VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

ZÁTĚŽOVÝ GENERÁTOR ZPRÁV DLMS/COSEM
DLMS/COSEM LOAD GENERATOR

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **David Kohout**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Tomáš Lieskovan**
SUPERVISOR

BRNO 2019

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní obor **Informační bezpečnost**
Ústav telekomunikací

Student: David Kohout ***ID:*** 195823
Ročník: 3 ***Akademický rok:*** 2018/19

NÁZEV TÉMATU:

Zátěžový generátor zpráv DLMS/COSEM

POKYNY PRO VYPRACOVÁNÍ:

Téma je zaměřeno na realizaci generátoru zpráv DLMS/COSEM využívaných v energetice (smart metering). V bakalářské práci se student seznámí s protokolem DLMS/COSEM, realizuje analyzátor a zátěžový generátor DLMS/COSEM komunikace. Své řešení student otestuje na laboratorní topologii.

DOPORUČENÁ LITERATURA:

[1] DLMS/COSEM Architecture and Protocols: EXCERPT FROM Companion Specification for Energy Metering [online]. [cit. 2018-09-13]. Dostupné z: http://dlms.com/documents/Green_Book_Ed_8-3_Excerpt.pdf

[2] Gurux.DLMS | Gurux for DLMS smart meters. [online]. [cit. 2018-09-13]. Dostupné z: http://www.gurux.fi/Gurux.DLMS

Termín zadání: 1.2.2019 ***Termín odevzdání:*** 27.5.2019

Vedoucí práce: Ing. Tomáš Lieskovan
Konzultant:

prof. Ing. Jiří Mišurec, CSc.
předseda oborové rady

ABSTRAKT

Práce se zaměřuje na problematiku Smart Meteringu, konkrétněji specifikaci DLMS/-COSEM. V práci jsou popsány nejdůležitější části standardu a jeho využití v chytrých sítích. Nejprve je rozebrán protokol DLMS, který zajišťuje vše týkající se komunikace mezi klienty a servery. Dále je popsán objektový model COSEM, který obsahuje jednotlivé objekty OBIS. Ukazuje možnosti odkazování se na objekty a nejdůležitější objekty popisuje.

V části analyzátor provozu je ukázán možný způsob analýzy provozu a DLMS zpráv. Analýza je poté využita pro sledování chování zařízení a generátoru. Popsána je také topologie sítě včetně přítomných zařízeních. Tato zařízení byla následně využita k vývoji a testování zátěžového generátoru DLMS zpráv.

Vytvořená aplikace je následně popsána z pohledu funkčnosti i zdrojového kódu. Během práce jsou popsány zjištěné nedostatky zařízení, včetně jejich chování při zatížení za použití vytvořeného zátěžového generátoru.

KLÍČOVÁ SLOVA

DLMS, COSEM, OBIS, Smart meter, zátěžový generátor, elektroměr

ABSTRACT

This work is focused on Smart Metering, specifically the DLMS/COSEM specification. The paper describes most important parts of the standard and its application in smart grids. At first the DLMS protocol is examined. It is responsible for everything related in communication between clients and servers. Next described think is object model COSEM and its objects called OBIS. It shows how can we reference to these objects.

In the second part – Traffic analyser are shown options of analysis of DLMS messages. This is after used for monitoring purposes. There is also described network topology including laboratory equipment. All devices were used for developing and testing the load generator of DLMS messages.

App, which was created for this work is then described. The paper also assesses found problems in security and implementation of DLMS protocol.

KEYWORDS

DLMS, COSEM, OBIS, Smart meter, load generator, electricity meter

KOHOUT, David. *Zátěžový generátor zpráv DLMS/COSEM*. Brno, 2019, 59 s. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce: Ing. Tomáš Lieskovan

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Zátěžový generátor zpráv DLMS/COSEM“ jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu bakalářské práce panu Ing. Tomáši Lieskovanovi za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Brno ....

podpis autora

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů

Úvod

V dnešním světě se stále více prvků každodenního života připojuje k internetu. Díky tomu máme aktuální přehled o situaci v domě, zemi i ve světě. Dále toto propojování senzorů, měřících a dalších zařízení umožňuje adaptaci, řízení a úpravu navazujících systémů, jako je například úprava cyklu semaforů na křižovatkách podle provozu nebo řízení výroby energie podle aktuální spotřeby, Případně ovládání různých prvků sítě pro co nejefektivnější využití zdrojů. Eventuálně se na řízení distribuční soustavy mohou částečně podílet i chytré elektroměry s využitím DLMS.

DLMS (Device Language Message Specification) je soubor standardů, které jsou vyvíjeny a spravovány asociací DLMS User Association, zkráceně DLMS UA [1]. Tyto specifikace a standardy byly také přejaty Mezinárodní elektrotechnickou komisí IEC, pod označením IEC 62056 [2]. V České republice byla norma přejata pod označením ČSN EN 62056 [3] a v podstatě se jedná o IEC 62056.

Objektový model COSEM (Companion Specification for Energy Metering, česky Společná specifikace pro měření energií) nám specifikuje modelování objektů, pro přístup k měřícím zařízením.

DLMS UA vyvinula standard a nyní se stará o údržbu standardu, registraci společností a certifikaci zařízení. Mottem společnosti a samotné specifikace je věta:

„We speak the same language.“

Čímž se odkazují na samotný standard, který se snaží o jednotný systém měření.

V této práci jsou popsány nejdůležitější části specifikace. Získané znalosti byly použity na vytvoření zátěžového generátoru DLMS zpráv. Generátor byl vytvořen pro zařízení přítomná v laboratoři, na kterých byl i testován. Testovaná zařízení však spadají pod Non-disclosure agreement (NDA). Tato smlouva neumožňuje zveřejňovat názvy zařízení, principy fungování, použité technologie ani vizuální podobu zařízení. Z tohoto důvodu jsou zařízení označovaná pouze jako koncentrátor, elektroměr atp.

1 Využití DLMS/COSEM

S příchodem a rozmachem internetu je jen otázkou času, než se k internetu připojí úplně každé zařízení. Proto ani v měření energií nezaostáváme a začínají se pomalu, ale jistě využívat tzv. smart metery. Nemusí se jednat pouze o elektroměry, DLMS specifikuje také objekty i pro další energie (voda, plyn, topení . . . viz tabulka 4.3).

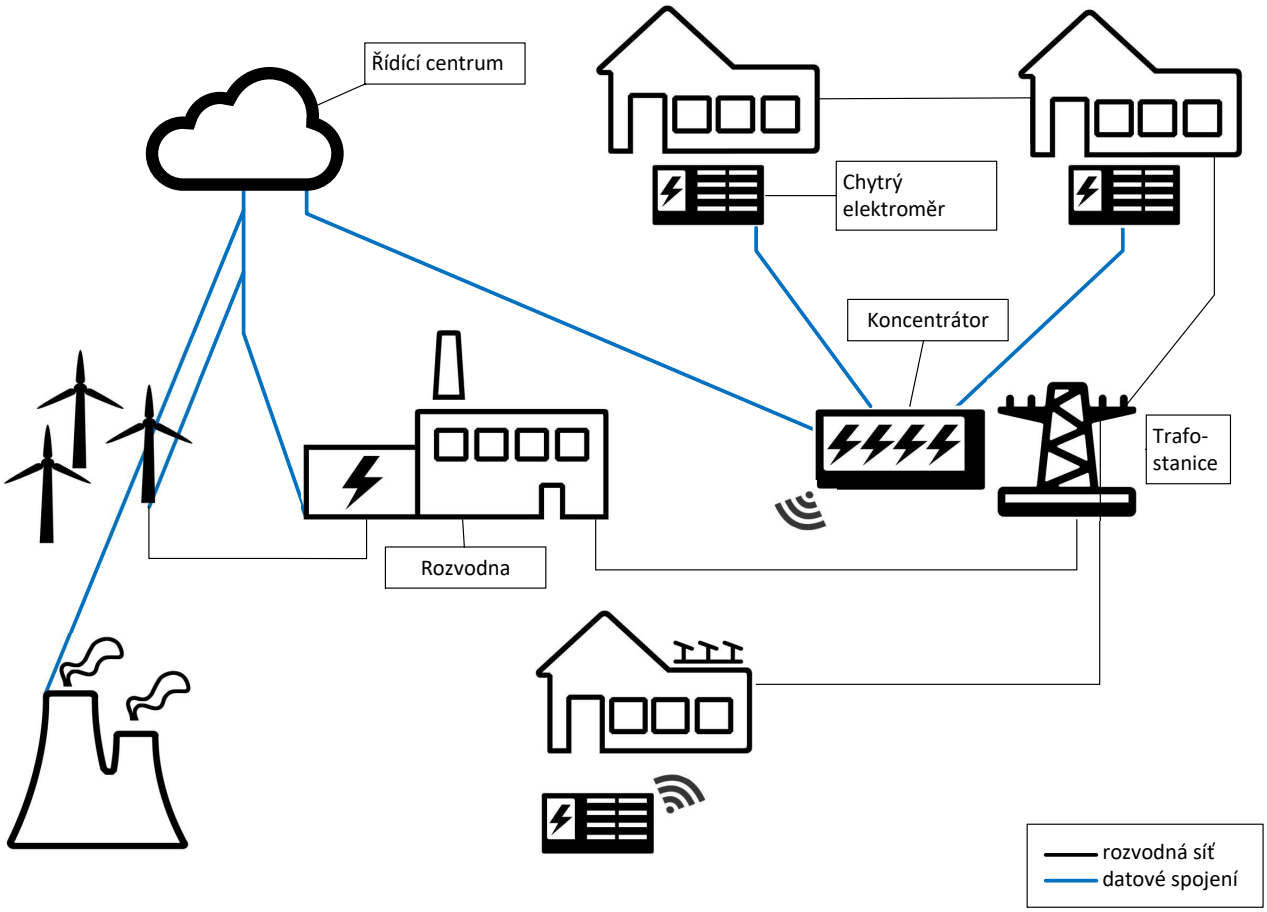
Práce se především zaměřuje na elektroměry, které jsou nejrozšířenější, především díky tomu, že u dalších měřících přístrojů nemusí být elektřina na dosah. Přivézt elektřinu a případně i data k ostatním měřícím přístrojům se zatím nevyplácí. Při stavbě, či přestavbě infrastruktury je potřeba myslet do budoucnosti a připravit tak infrastrukturu na snadné rozšíření o nové prvky.

Tyto smart metery neboli chytré elektroměry mohou obousměrně komunikovat a tím umožňují lepší řízení výroby elektrické energie, usnadňují odečty, mohou pomoci s identifikací problémů v sítích a také mohou rozpoznat černé odběry. Odečty mohou probíhat jednou i několikrát měsíčně nebo ještě častěji. Například u malovýrobců s vlastní výrobnou (např. fotovoltaické panely) na domě, se odečty opakují zhruba každých 15 minut.

Pokud by každé odběrné místo, každou minutou, komunikovalo s řídícím centrem, jednalo by se o ohromný datový tok. Proto existují tzv. koncentrátory, které snižují datovou náročnost. Na jeden koncentrátor mohou zasílat data desítky až stovky zařízení. Teoreticky každý panelový dům nebo každá ulice může být napojena na jediný, který může být umístěný v trafostanici. Dnes si společnosti spravující elektrickou síť zavadí spolu s novými kabely i vlastní optiku, především do nově stavěných trafostanic. To zejména pro jejich správu a řízení. Optika v takovém místě tedy poskytuje dostatečnou datovou propustnost pro komunikaci většího počtu koncentrátorů s řídícím centrem. Nakonec spolu komunikují řídící centra a elektrárny a tím dokáží spravovat celou síť. Schéma možné sítě je vyobrazeno na obrázku 1.1.

Síť, kde výrobci i spotřebitelé využívají chytré měření se dá označovat jako „Smart Grid“. Tyto typy sítí mohou automaticky, inteligentně a efektivně řídit celou distribuční soustavu. V České republice existuje projekt, který se snaží dosáhnout automatizace a online monitorování. Jedná se o tzv. Smart region Vrchlabí [4].

Hromadný přechod na smart metery bohužel není tak snadný. Například společnost ČEZ, u zmiňovaného projektu ve Vrchlabí, komentuje situaci následovně: „*Benefity vyplývající z používání chytrých elektroměrů, zatím nevyvažují vysokou investici a nepřinášejí dostatečnou motivaci pro spotřebitele, aby tohoto způsobu chytrého měření využívali.*“ [4]



Obr. 1.1: Schéma chytré sítě

Další překážkou v nasazování chytrých elektroměrů kromě ceny je již existence prvku „chytré“ sítě. Tím je hojně využívaný systém HDO (hromadné dálkové ovládání). HDO umožňuje pomocí signálů, vysílaných přímo po elektrických rozvodech, zasílat spotřebičům spínací povely. Zároveň se tyto signály využívají na přepínání mezi tarify (např. nízká sazba za elektřinu tzv. noční proud).

Určitě není na škodu při výměně starých elektroměrů využívat již moderních chytrých elektroměrů. I za cenu, že by se data vysílala pouze ke spotřebiteli. Možné by tak bylo využití informací z elektroměrů pro aplikace v mobilních telefonech, aby člověk mohl mít přehled, jak moc elektřiny spotřebuje. Dále by se výstup aktuální spotřeby/výroby dal využít pro řízení domácnosti (topení, ohřívání vody, běh filtrace apod.).

2 DLMS/COSEM

Device Language Message Specification a Companion Specification for Energy Metering, zkráceně DLMS/COSEM, specifikuje model rozhraní a komunikační protokoly pro výměnu dat mezi měřícími zařízeními. Celá specifikace byla vytvořena společností DLMS UA [1] a je rozdělena do 4 částí. Pro snazší zapamatování jsou tyto části rozděleny do barevně označených knih tzv. Coloured Books [5, 6]. Jmenovitě Green Book [7], Blue Book [8], Yellow Book [9] a White Book. Green Book popisuje architekturu a protokoly, Blue Book popisuje COSEM objekty a OBIS (OBject Identification System), Yellow Book popisuje postup testování a White Book je slovník pojmů.

Samotný princip specifikace se dělí na 3 body: [7]

1. Modelling (modelování)
    - Tvorba objektů
    - Zahrnuto v Blue book [8]
2. Messaging (tvorba zpráv)
    - Překlad objektů na posílatelné zprávy (APDU)
    - Zahrnuto v Green book [7]
3. Transporting (přenos)
    - Přenos zpráv po komunikačních kanálech
    - Částečně v Green book [7]
    - Využití nižších vrstev (např. TCP/IP, NB-IoT, LTE, GPRS)

Aplikační vrstva DLMS/COSEM AL (kapitola 3.6) specifikuje služby pro navázání spojení mezi klientem a serverem (případně více servery), dále poskytuje služby pro přístup k atributům a metodám COSEM objektů.

Stále vetší rozšíření chytrých systémů pro měření vyžaduje zabezpečení přenášených informací k ochraně soukromí zákazníků. DLMS/COSEM obsahuje i zabudované bezpečnostní mechanizmy. Poskytuje tak služby pro identifikaci a autentizaci klientů a serverů, dále přiděluje práva pro přístup ke COSEM objektům. Vše se odehrává v rámci prvního kroku spojení tzv. Aplication Associations (AA). Přenášené zprávy APDU (Application Protocol Data Unit) mohou být také šifrované a tím přenášené zprávy zabezpečují.

Specifikace obsahuje i rozšíření o další kryptografické algoritmy. Jako například šifrování už samotných objektů ještě předtím, než jsou vytvořeny APDU, symetrické a asymetrické algoritmy pro autentizaci, šifrování a digitální podpisy. [7]

3 DLMS

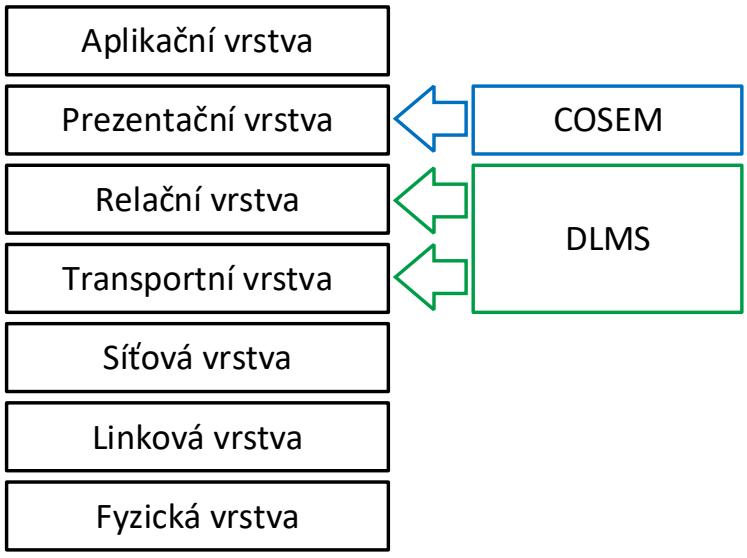
Protokol DLMS se stará o překlad objektů na posílatelné zprávy. Dále zajišťuje navazování a zabezpečování komunikace mezi klientem a serverem. Nejdůležitější pro samotnou komunikaci je však přenos informací do nižších vrstev, které se starají o samotný přenos.

3.1 Komunikační model

Referenční model OSI/ISO se využívá pro popis fází komunikace mezi zařízeními z hlediska přenosu dat. Skládá se ze sedmi vrstev, kde navenek jednotlivé vrstvy komunikují se stejnou vrstvou mezi zařízeními.

Výměna dat mezi měřiči a sběrným systémem je založena na modelu klient/server, kde roli serveru hraje většinou elektroměr (případně jiné měřící zařízení) a role klienta připadá na sběrný systém. Klient posílá žádost a server na ni odesílá patřičnou odpověď. Případně může server informovat klienta o událostech nebo odeslat data podle přednastavených podmínek bez předchozí žádosti ze strany klienta.

DLMS/COSEM můžeme umístit na model OSI/ISO následujícím způsobem, který je znázorněný na obrázku 3.1. Objektový model COSEM překládá a formátuje přenášená data, proto jej lze umístit na prezentační vrstvu. Kdežto protokol DLMS se stará o dialog a výměnu dat mezi dvěma zařízeními, tudíž jej lze umístit na relační a transportní vrstvu. Zbytek komunikace na nižších vrstvách již není zahrnut ve specifikaci DLMS/COSEM a pro transport dat se využívají různé způsoby. [7]



Obr. 3.1: DLMS/COSEM na modelu OSI/ISO

3.2 Adresace a jména

Adresace a jména jsou důležitým aspektem v komunikačních systémech. Jména mohou být mapována na adresy – proces „bindování“.

Jména entit (klientů, serverů a třetích stran) se odvíjejí od jejich systémových jmen. Tyto jména by měla být jedinečná a permanentně přidělená. Systémové jméno se označuje jako `Sys-T` [7], které je 8 oktetů dlouhé a jedinečné. První 3 oktety obsahují třípísmenné ID výrobce. Zbylých 5 oktetů se vytváří tak, aby zajišťovaly jedinečnost.

ID společnosti se označuje jako FLAG ID. Jsou udělována DLMS UA ve spolupráci s FLAG Association. Znaky jsou pouze velké a využívá se zde pouze základní latinka, což je 26 písmen. Tím získáváme $26^3 = 17\,576$ možností. V červenci 2018 bylo zaregistrovaných 1 303 FLAG ID [10]. S rostoucím počtem výrobců a využití se ID přidělují velmi šetrně. Zároveň je snaha zkratku co nejlépe přizpůsobit pro název firmy např. AEG a ABB jsou třípísmenné názvy firmem a dostaly identickou zkratku s názvem.

```
ABC = aBC, AbC, ABc, abC, aBc, Abc, abc.
```

Adresa každého zařízení je závislá na druhu použitého komunikačního profilu. Může to tedy být například telefonní číslo, MAC adresa, IP adresa nebo jejich různá kombinace.

Fyzická adresa zařízení může být předkonfigurovaná nebo může být přidělena během registračního procesu, který také váže adresu na `Sys-T`. Každý klient a server v rámci DLMS/COSEM je svázán s Service Access Pointem (SAP). SAP zde plní podpůrnou funkci DLMS AL (kapitola 3.6) a také vychází z použitého komunikačního profilu. Na straně serveru je svázání SAP a `Sys-T` označováno jako „SAP Assignment“.

3.3 Orientace spojení

Komunikační relace se skládá ze 3 fází:

- První se jmenuje Aplication Associations (AA), dochází zde k navázání spojení mezi klientem a serverem.
- V druhé fázi dochází k výměně zpráv.
- V poslední fázi na konec dochází k rozpojení spojení – zrušení AA.

V případě použití jednodušších zařízení je možné využít jednosměrnou, multicastovou, či broadcastovou komunikaci. Zde se využívá přednastavené AA. Tudíž se nemusí spojení pracně vytvářet a komunikace obsahuje jen fázi výměny zpráv. [7]

3.4 Aplication Associations

Aplication Associations jsou logické spojení mezi klientem a serverem. Mohou být vytvořené na žádost, nebo mohou být předkonfigurované. Zařízení může podporovat i více aktivních AA. Spojení navrhuje klient a server jej potvrzuje tehdy, pokud:

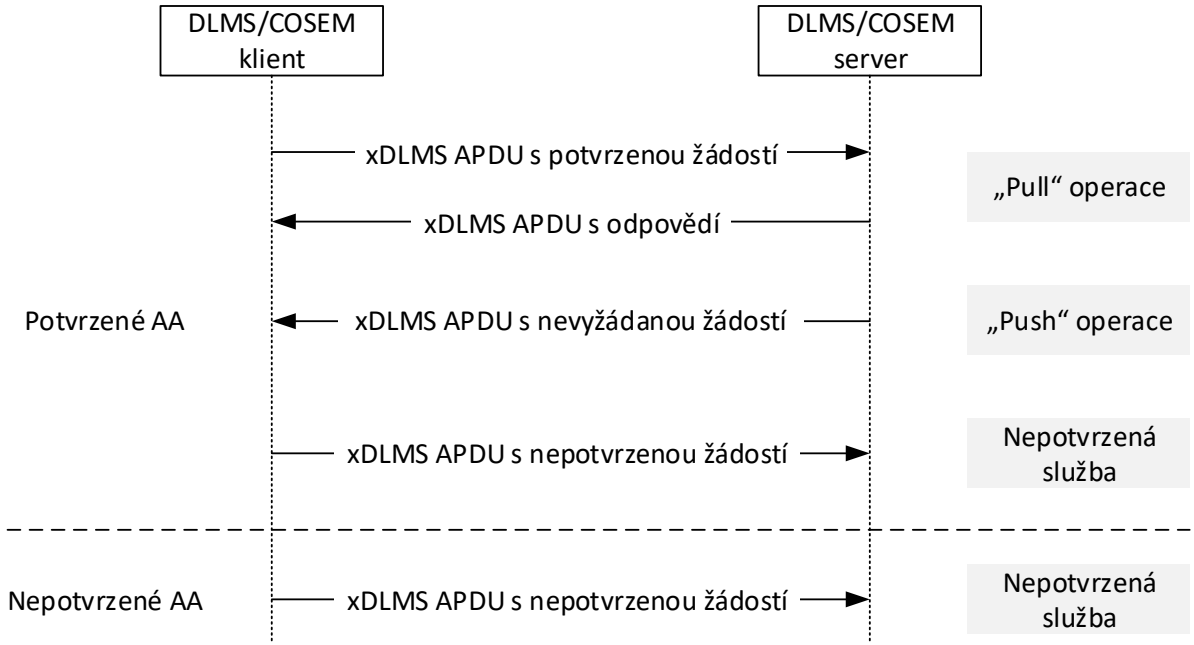
- uživatel i klientské zařízení je známo na serveru,
- aplikační kontext je přijatelný,
- autentizační mechanizmy navrhované klientem odpovídají serverovým.

Může dojít i k nepotvrzenému AA, které se většinou využívá pro broadcast komunikaci klienta s více servery. Během AA dochází k mapování LN/SN jmen a určují specifická přístupová práva ke COSEM objektům. [7]

Aplikační kontext určuje:

- zdali se bude používat šifrování,
- použité odkazování SN nebo LN,
- syntax přenosu,
- skupinu ASE (Application Service Elements).

3.5 Možné druhy spojení



Obr. 3.2: Možné druhy spojení

Průběh spojení lze vidět na obrázku 3.2. [7]

V **potvrzeném** AA:

- Pull operace – Klient zasílá potvrzenou žádost, server na ni odpovídá
- Push operace – Server zasílá nevyžádanou žádost klientovi (Informace o stavu, upozornění, plánované vyúčtování . . .)

U **potvrzeného** i **nepotvrzeného** AA může ještě proběhnout:

- Nepotvrzená služba – Klient může zaslat nepotvrzenou žádost, server na ni neodpovídá

3.6 Aplikační vrstva DLMS/COSEM

Struktura aplikačních vrstev DLMS [7] lze vidět na obrázku 3.3. Hlavní součástí aplikační vrstvy je takzvaný Application Service Object (ASO), který zpřístupňuje uživateli služby a aplikační procesy (AP). ASO se skládá ze 3 částí společných pro klienta i server:

- ACSE – Association Control Service Element, poskytuje služby pro navázání/ukončení spojení pomocí AA.
- xDLMS ASE – rozšířený DLMS Application Service Element, poskytuje služby pro transport dat mezi aplikačními procesy.
- CF – řídící funkce, specifikuje, jak se bude ASO chovat k ostatním službám.

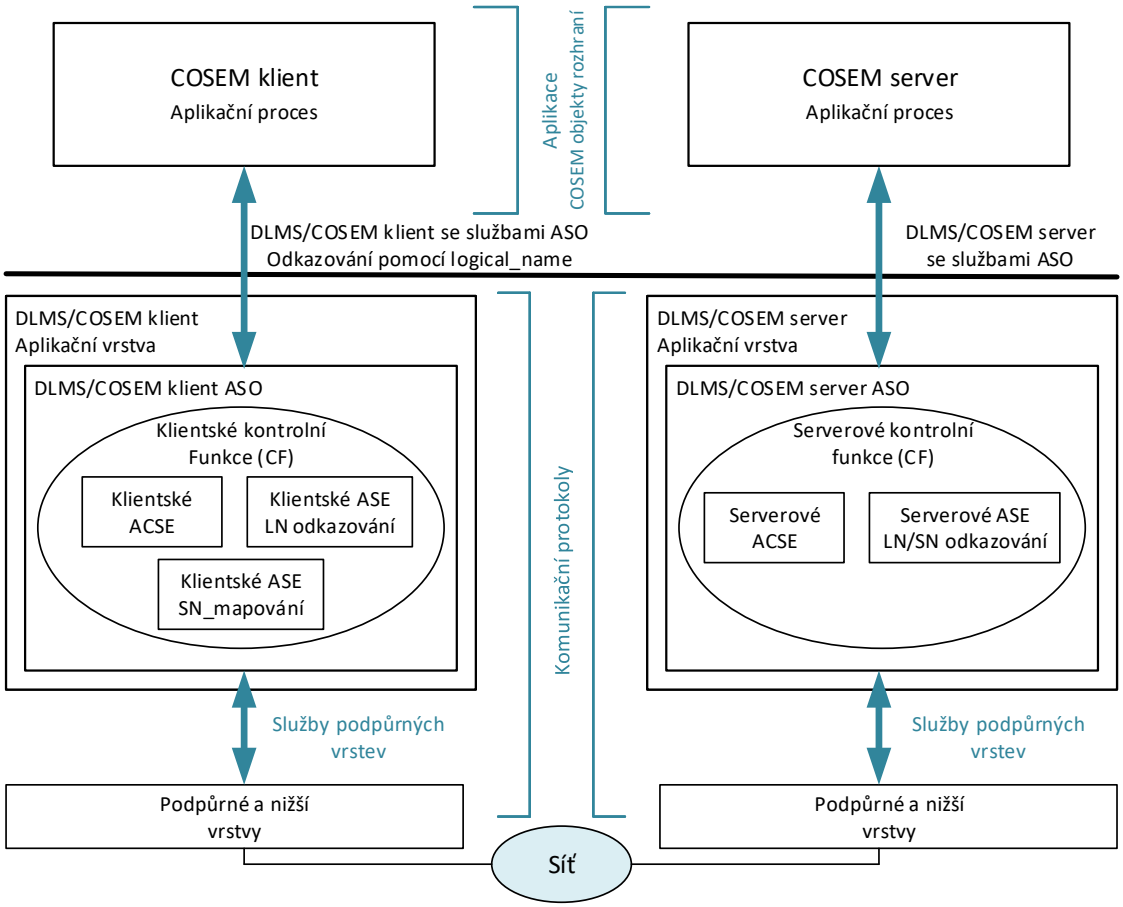
Na straně klienta je i čtvrtá funkce a to `SN_Mapper` pro mapování LN na SN názvy. Dále se ASO účastní na prezentační vrstvě OSI, kde provádí následující:

- Kódování a dekódování APDU zpráv
- Provádění kompresních služeb
- Vytváření a ověřování kryptografických zabezpečení
- Případně generování XML dokumentů

3.6.1 Association Control Service Element

ACSE – poskytuje následující služby pro navázání AA [7]:

- COSEM-OPEN – slouží pro navázání AA podle stanovených hodnot z aplikačního kontextu, bezpečnostních mechanizmů a dalších parametrů.
  - V potvrzeném AA dochází ke spojení pomocí výměny zpráv s využitím právě COSEM-OPEN služeb. Ty se mezi klientem a serverem snaží určit kontext.



Obr. 3.3: Struktura aplikačních vrstev DLMS/COSEM[7]

  - V nepotvrzeném AA se spojení navazuje pomocí zaslané zprávy OPEN od klienta k serveru. Využívá se zde kontextu, který klient považuje za správný.
  - U předspojeného AA se COSEM-OPEN nevyužívá. Klient i server by již měli znát kontext, který budou používat. Toto spojení může být potvrzené i nepotvrzené.
- COSEM-RELEASE – slouží pro ukončení AA spojení. Pokud je ukončení úspěšné, nedochází ke ztrátě informací při přenosu. Předspojené AA nemůže být ukončeno.
- COSEM-ABORT – slouží k vynucenému ukončení v případě nějaké kritické chyby. Může dojít ke ztrátě dat.

3.6.2 Prvky aplikačních služeb – ASE

Služby pro odkazování

Odkazování umožňuje přístup ke COSEM objektům. Může obsahovat různá rozšíření i se zachováním zpětné kompatibility se základní verzí. Jelikož se k objektům dá přistupovat pomocí LN nebo SN [7, 11], tak je potřeba specifikovat každý druh volání zvlášť. Klient většinou používá odkazování LN, server může používat buď jen jeden, nebo oba druhy. K případnému překladu dochází na straně klienta v rámci služby `SN_Mapper`.

Při LN odkazování se využívají následující služby [7]:

- GET – Službu využívá klient pro získání atributů z COSEM objektů serveru. Používá se zde model žádost-odpověď. Využívá PDU pro žádost typu `get-request` a pro odpověď `get-response` [12].
- SET – Služba využívaná klientem pro změnu nebo nastavení hodnot u určeného atributu COSEM objektu. PDU typu `set-request` a `set-response`.
- ACTION – Slouží pro zavolání jedné nebo více metod. Pokud je metoda návratová, server zpátky zašle návratovou hodnotu. PDU typu `action-request` a `action-response`.
- ACCESS – Jednotná služba, která umožňuje přístup k více atributům a metodám v jediné žádosti.

Metody GET, SET a ACCESS mohou volat i tzv. „`Attribute_0`“ [7]. Všechny atributy objektů jsou však indexovány od 1, kde prvním atributem je logické jméno COSEM objektu. Tento `attribute_0` odkazuje na všechny atributy v objektu. Jeho využití bohužel není ve veřejné specifikaci uvedeno.

Služby u SN odkazování [7]:

- Read – Služba s funkčností jako GET, ACTION a ACCESS dohromady. Umožňuje klientovi zažádat server o vrácení jedné či více hodnot nebo pro zavolání metod s předpokládaným návratovým parametrem. PDU typu `readRequest` a `readResponse`.
- Write – Služba podobná službě SET. Umožňuje změnu nebo nastavení hodnot atributů. Dále může volat nenávratové metody. PDU typu `writeRequest` a `writeResponse`.
- UnconfirmedWrite – Nepotvrzená služba, jinak stejná jako služba write. PDU typu `unconfirmedWriteRequest` a `unconfirmedWriteResponse`.

Identifikace odpovědí

Jelikož se pro komunikaci využívá model klient/server, žádosti zasílá klient a odpovědi server, tak je možné, že klient zašle více žádostí po sobě dříve, než dojde odpověď. Aby bylo možné rozeznat, která odpověď patří, které žádosti, zavádí se pojem `Invoke_Id` [7]. Toto ID je uvedeno na žádosti a zasílá se zpět s odpovědí.

Parametr `Invoke_Id` přiděluje žádostem klient, díky tomu by každé ID mělo být jiné. Server poté jen parametr zkopíruje do odpovídající odpovědi.

V případě využití SN odkazování není tato funkce dostupná.

Úprava zpráv a přenos dlouhých zpráv

Posílané příkazy a zprávy mohou být různými způsoby upravovány a komprimovány. Mohou se kódovat/dekódovat případně se mohou aplikovat/ověřovat/odebírat kryptografické ochrany. Následně jsou přenášeny pomocí xDLMS APDU. Pokud se stane, že výsledná APDU je větší než maximální velikost PDU, je nutné tuto zprávu upravit (velikost PDU se určuje při navazování spojení).

Existují 2 možné metody úprav:

- SBT (service-specific block transfer [7]) – mechanismus dostupný pouze pro GET, SET, ACTION, Read a Write služby. Tyto služby se dají ještě před úpravou rozdělit na jednotlivé příkazy, jelikož se mohou dotazovat na více proměnných, či volat vícero metod. Rozdělené zprávy se pak kódují zvlášť a jejich velikost tak odpovídá limitu pro vložení do APDU.
- GBT (general block transfer [7]) – umožňuje rozdělení APDU do bloků. Takto rozdělené zprávy se odesílají a druhá strana může příjem buď jednotlivě, nebo jako celek potvrdit. V případě nepotvrzení přijetí se daný blok zašle opakovaně.

3.7 Informační bezpečnost v DLMS/COSEM

Pro zajištění bezpečnosti a důvěrnosti se v DLMS/COSEM využívá řada mechanizmů. Patří k nim identifikace, autentizace, autorizace a šifrování [7].

Většina algoritmů pro zabezpečení může být implementována samotnými výrobci. V případě využití těchto proprietárních funkcí, je komunikace s využitím těchto mechanizmů možná pouze mezi zařízeními stejného výrobce.

3.7.1 Autentizace

Autentizace je pro bezpečnost velmi důležitým prvkem. Cílem autentizace je ověřit identitu přistupujícího. V DLMS je vykonávána během vytváření spojení – AA.

Identitu lze ověřit ze strany klienta, serveru nebo vzájemně. Navíc může server požadovat i identifikaci uživatele na straně klienta. V nepotvrzeném AA se klient autentizuje sám. V přednavázaném AA autentizace není možná. Po úspěšné autentizaci může ke zdrojům serveru (COSEM atributy a metody objektů) přistupovat klient. [7]

DLMS specifikuje tři druhy autentizace podle zabezpečení:

- Lowest Level Security (bez zabezpečení) [7]
  - Slouží pro získání základních informací o serveru.
- Low Level Security (LLS) [7]
  - Klient se autentizuje poskytnutím hesla, které je známo serveru.
  - Heslo je zasíláno serveru v rámci žádosti COSEM-OPEN.
  - Pokud je heslo správné, AA spojení je navázáno. Pokud ne, tak dojde k odpojení.
- High Level Security (HLS) [7]
  - Pro úspěšné navázání se klient i server musí vzájemně autentizovat.
  - Využívá se zde model výzva-odpověď, proto je ověření čtyřcestné.
  - Klient zasílá žádost společně s výzvou. Server zasílá zpět odpověď společně s vlastní výzvou. Následně klient výzvu zpracuje a zasílá odpověď. To samé provede i server. Pokud jsou odpovědi na výzvy správné, spojení je navázáno.
  - Využít lze algoritmy MD5, SHA-1, GMAC, SHA-256 a EC-DSA.

Jaký způsob zabezpečení se použije, určuje hodnota proměnné `mechanism_id`. Hodnoty, kterých může `mechanism_id` nabývat jsou uvedeny v tabulce 3.1.

3.7.2 Zabezpečení zpráv APDU

První APDU zpráva je sestavena bez zabezpečení. Díky ní klient získá parametr `security_options` [7]. Tento parametr určuje, jaká šifrovací primitiva se pro šifrování využijí. Zároveň poskytuje další informace nezbytné pro vytvoření zabezpečeného spojení. Po získání těchto informací se již sestavují šifrované APDU.

V rámci šifrování APDU se určují přístupová práva složená z 8 bitů označovaných jako `unsigned8`. Oprávnění se vztahují na přístup, žádosti a odpovědi. Význam jednotlivých bitů je uveden v tabulce 3.2.

Tab. 3.1: ID autentizačních mechanizmů [7]

Druh autentizace	**Mechanism_id**
Bez zabezpečení	0
LLS	1
HLS	2
HLS s využitím MD5	3
HLS s využitím SHA-1	4
HLS s využitím GMAC	5
HLS s využitím SHA-256	6
HLS s využitím EC-DSA	7

Tab. 3.2: Význam bitů s přístupovými právy [7]

Bit	**Přístup k atributům**	**Přístup k metodám**
0	Čtení	Přístup
1	Zápis	-nevyužito-
2	Autentizovaná žádost	Autentizovaná žádost
3	Šifrovaná žádost	Šifrovaná žádost
4	Digitálně podepsaná žádost	Digitálně podepsaná žádost
5	Autentizovaná odpověď	Autentizovaná odpověď
6	Šifrovaná odpověď	Šifrovaná odpověď
7	Digitálně podepsaná odpověď	Digitálně podepsaná odpověď

Dekadicky vyjádřené příklady:

- 3 = čtení a zápis
- 6 = zápis s autentizovanou žádostí
- 255 = čtení a zápis obsahující autentizované, šifrované a digitálně podepsané žádosti i odpovědi

Pro šifrování zpráv se používá symetrický algoritmus AES v GCM módu [7]. Jako inicializační vektor se používá `Sys-T`, který je upravován pomocí některého čítače (například přičítání nebo násobení určitou hodnotou). Výsledná zašifrovaná zpráva obsahuje zabezpečovací byte, čítač (použitý pro úpravu `Sys-T`), zašifrovaný obsah a v případě šifrování s autentizací se na konec zprávy přidává autentizační klíč [13].

Bližší informace o použitých klíčích, algoritmech a detailech nejsou veřejně přístupné.

4 COSEM

COSEM představuje objektově orientovaný přístup k objektům. Vytváří tedy rozhraní, díky kterým si mohou sběrné systémy a měřící zařízení vyměňovat data. Tím dosahují interoperability.

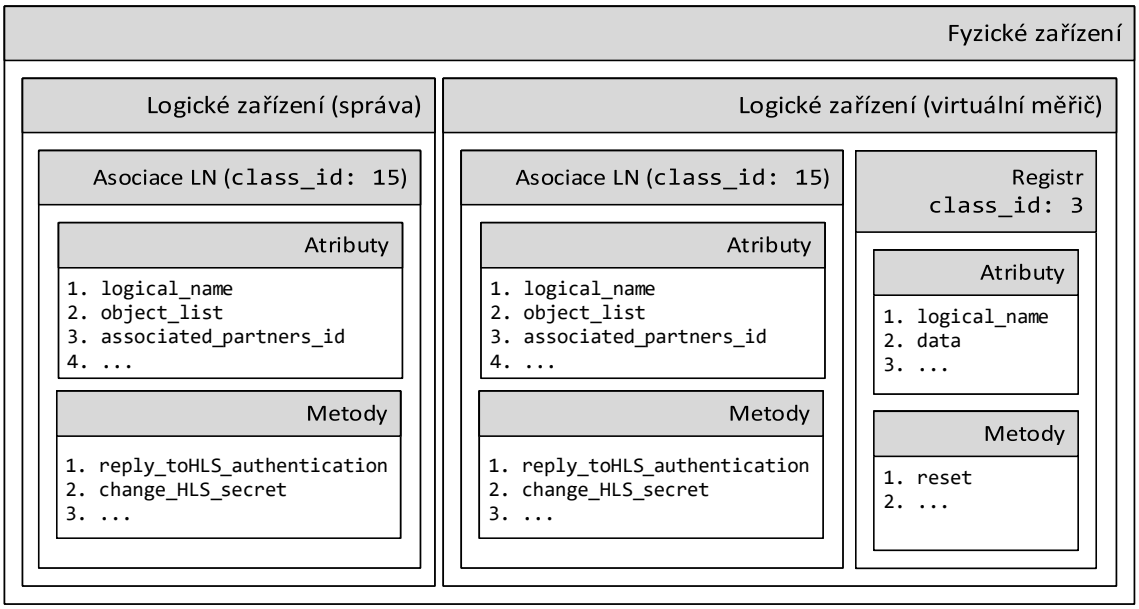
Jednotlivé objekty jsou sestaveny z metod a atributů. Charakteristiku objektu představují právě atributy, jejichž hodnoty mají vliv na celý objekt. Důležitým atributem každého objektu je `logical_name`, který hraje roli identifikátoru objektu. Podobné objekty jsou zobecněny v rozhraních (tj. třída určující povinné prvky kódu pro vlastní implementaci). Výrobci si tak mohou přidávat do objektů vlastní metody pro lepší funkci s jejich zařízením. Avšak musí zachovat základní objekty, aby bylo možné komunikovat mezi zařízeními různých výrobců.

Standard COSEM definuje přibližně 70 tříd rozhraní [8]. Rozhraní je definováno svým jménem, verzí a `class_id`. Každý atribut má definované jméno, datový typ, výchozí, minimální a maximální možnou hodnotu. Kromě jména mohou atributy obsahovat definované krátké jméno (`short_name`).

4.1 COSEM server model

COSEM server se dá rozdělit do 3 úrovní, vyobrazených na obrázku 4.1

1. Fyzické zařízení
2. Logické zařízení
3. Jednotlivé COSEM objekty



Obr. 4.1: Model COSEM zařízení [14]

4.1.1 COSEM logické zařízení

Logické zařízení obsahuje jednotlivé objekty. Každé fyzické zařízení může mít více logických objektů, ale minimálně by mělo obsahovat řídící logické zařízení. Adresování na logická zařízení je možné přes LDN (Logical Device Name). LDN lze získat zavoláním „SAP assignment“ (viz tabulka 4.1). LDN je velice podobné jménu u DLMS (kapitola 3.2), ale může být místo 8 až 16 oktetů dlouhé. První 3 oktety jsou opět přiděleny výrobci, zbytek si určuje výrobce sám.

Řídící logické zařízení je nutností pro správné fungování komunikace. Mělo by mít rezervovanou adresu, mělo by podporovat AA (kapitola 3.4) s veřejným klientem s co nejnižším stupněm autentizace, případně úplně bez ní. Hlavní funkcí řídícího zařízení je poskytovat vnitřní strukturu celého zařízení a spravovat události v serveru. [8]

4.2 Odkazování na objekty COSEM

Pro odkazování na objekty se využívají 2 druhy odkazování [7, 8]:

1. Delší `logical_name` (LN)
2. Kratší `short_name` (SN)

V případě LN jsou objekty volány přímo přes identifikátor COSEM objektu, pod který patří. Skládají se tedy ze šesti hodnot definovaných podle systému OBIS (viz kapitola 4.4). Odkazování se na metodu objektu je následující [8]: `class_id`, jméno metody a index metody (`method_index`). Pokud se jedná o atribut, volá se `class_id`, jméno atributu a index atributu (`attribute_index`). Tyto indexy jsou specifikovány v definici každého rozhraní.

SN se využívá u jednoduchých zařízení. Každý atribut i metoda objektu je identifikovatelná 13bitovým číslem. Pro jeho využití musí být objekty namapované na krátká jména. Při volání metody nebo atributu pomocí SN se přičítá určitá („offset“) hodnota k základnímu jménu objektu (např. Asociace SN má jméno hodnotu 0xFA00 a pokud se přičte 0x08, získá se tak seznam objektů). Volání je tedy jednodušší, nemusí se uvádět další parametry. Může ovšem nastat situace, kdy výrobce používá jiné hodnoty (přiřazení SN jiným LN) a díky tomu je nemožná správná komunikace mezi zařízeními různých výrobců. Proto specifikace uvádí pouze hodnoty LN [8].

4.2.1 Speciální rezervovaná jména objektů

Pro využívání SN jmen, musí existovat základní seznam jmen tzv. `base_names` [8]. Existují 4 základní objekty, které jsou definované v tabulce 4.1. Nejdůležitějším objektem je „Asociace SN,“ který umožňuje získat všechny jména SN objektů na daném zařízení. Takto se získá objekt s názvem `object_list` obsahující pole struktur.

Každá položka struktury pak nese hodnoty:

- `logical_name` – LN jméno,
- `base_name` – SN jméno,
- `class_id` – číslo třídy rozhraní,
- `version` – verze rozhraní.

Tab. 4.1: Rezervovaná `base_names` pro odkazování přes SN [8]

`Base_name` (SN)	**COSEM objekt**
0xFA00	Asociace SN
0xFB00	Tabulka skriptů
0xFC00	Přidělení SAP
0xFD00	Data nebo Registr objektů s LDN

4.3 Přehled základních rozhraní – `class_id`

Následuje hrubý výběr důležitých rozhraní. V závorce je uvedeno `class_id` [8]

- Data (1)
  - Umožňuje vytvořit základní datové typy pro COSEM objekty.
- Registr (3-6)
  - Rozšiřuje třídu Data o jednotku a měřítko. Případně obsahuje čas záznamu, průměrné hodnoty, či přepínání mezi tarify.
  - Má více typů (proto více ID). Extended, Demand, Activation.
- Obecný profil (7)
  - Umožňuje uchovávání, přístup a řazení shluků objektů. Obsahuje atribut *buffer*, kde uchovává všechna data.
- Hodiny (8)
- Asociace SN (12)
- Asociace LN (15)
- Přidělení SAP (17)
- Ochrana dat (30)
- Nastavení Push operací (40)
  - Po nastavení, server zasílá data klientovy bez žádosti. (Odesílají se: podle plánu, podle překročení určité hodnoty, vyvoláním nějakou událostí)
- Nastavení TCP-UDP (41)
- IPv4 (42)
- IPv6 (48)
- Kompaktní data (62)
  - Mohou obsahovat denní vyúčtování, logy, diagnostiku nebo poplach.

- Nastavení zabezpečení (64)
- Správa senzoru (67)
  - Zařízení může obsahovat tlakový spínač. Při narušení ohlásí poplach.
- ZigBee (101-105)
- Nastavení funkcí – CF (122)
  - Slouží pro zapínání/vypínání funkcí většinou ovládaných časem.
- LTE monitoring (151)

4.4 OBIS

OBIS je zkratka pro OBject Identification System. Poskytuje nám jedinečné identifikátory pro všechna data v měřících zařízeních, včetně naměřených hodnot. Dále obsahuje i abstraktní hodnoty pro konfiguraci a získávání informací o chování celého zařízení.

Všechny ID, která OBIS definuje se používají pro [8]:

- logická jména tříd rozhraní a objektů,
- přenášená data přes komunikační rozhraní,
- zobrazená data na zařízení.

Kódy jsou rozděleny do 6 skupin (A–F), zobrazených v tabulce 4.2

Hodnoty každé skupiny mohou dosahovat hodnot 0–255. Obsahují implementované hodnoty, rezervované, případně volné pro jednotlivé výrobce. Hodnoty pro výrobce většinou mají rozsah 128–199 nebo 128–254, kromě skupiny A a D, které jsou kompletně rezervované.

První skupina A (tab. 4.3) je výchozí a každá další skupina je závislá na hodnotách předchozích skupin. Například pokud je hodnota skupiny C rovna 93, tak následující skupina ponese hodnoty společností. Bohužel v době vydání Blue book (19. 1. 2017 [8]) byla tato skupina ještě prázdná. V případě, že C=94, tak skupina D nese hodnoty specifické pro určitou zemi. Od země se odvíjí i skupiny E a F. Česká republika nese hodnotu 10, takže se nacházíme celkem vysoko. Navíc je zde ještě 5 neobsazených míst a tím pádem jsme 6. zemí v pořadí. První je Finsko s hodnotou 0. Výčet pár hodnot zemí: USA-1, Kanada-2, Srbsko-3, Rusko-7, Slovensko-42, Německo-49.

Tab. 4.2: Struktura OBIS kódů a využití jednotlivých skupin [8, 11]

Skupina	**Využití hodnot**
A	Popisuje typ energie, se kterou zařízení pracuje.
B	Udává kanál, ze kterého pochází měřená data. Důležité především v koncentrátorech. Hodnoty jsou nezávislé na skupině A.
C	Specifikuje druh měřených dat (napětí, proud, teplotu...)
D	Popisuje výsledky ze zpracování hodnot ze skupin A–C.
E	Doplňuje zpracování hodnot měření z předešlých skupin.
F	Může určovat historii dat nebo specifikovat předchozí skupiny.

Tab. 4.3: Hodnoty skupiny A [8]

Hodnota	**Využití**
0	Abstraktní
1	Elektřina
4	Akumulátor tepla
5	Chlazení
6	Topení
7	Plyn
8	Studená voda
9	Teplá voda
15	Ostatní média
Zbytek	Rezervovaný

4.4.1 Objekty pro elektřinu. Hodnota skupiny A = 1

Pro ukládání většiny hodnot se využívají rozhraní Data a Registr (a jeho pod části...). Jejich vzájemnou kombinací lze vytvářet maxima a minima, časové hodnoty, čítače, kumulativní a smluvní hodnoty [8].

Kódy skupiny C (A=1, B – různé)
Prvních 80 hodnot je rozděleno vždy do čtyř skupin, kde prvních 20 je suma fází a další 3 skupiny po 20ti uvádí jejich jednotlivé hodnoty. Například hodnoty 1, 21, 41 a 61 jsou vázané na stejnou informaci. Uvádí například napětí, proud, frekvenci a import/export činných a jalových výkonů.

Další hodnoty jsou například úhly, neutrální hodnoty, specifikace země a společnosti, objekty chyb a jiné.

Kódy skupiny D (A=1, B – různé, C vše kromě 0 a 93–99)
V této skupině jsou například průměry hodnot za účtovací období, různá maxima, minima, časové integrály, čítače a prahové hodnoty ...

Kódy skupiny E
Převážně se jedná o různé tarify (specifikace neuvádí, o jaké tarify se jedná). Momentálně obsahuje 64 možných tarifů. Dále může specifikovat až 120 harmonických frekvencí nebo definovat úhly mezi fázemi zvlášť pro proud a napětí.

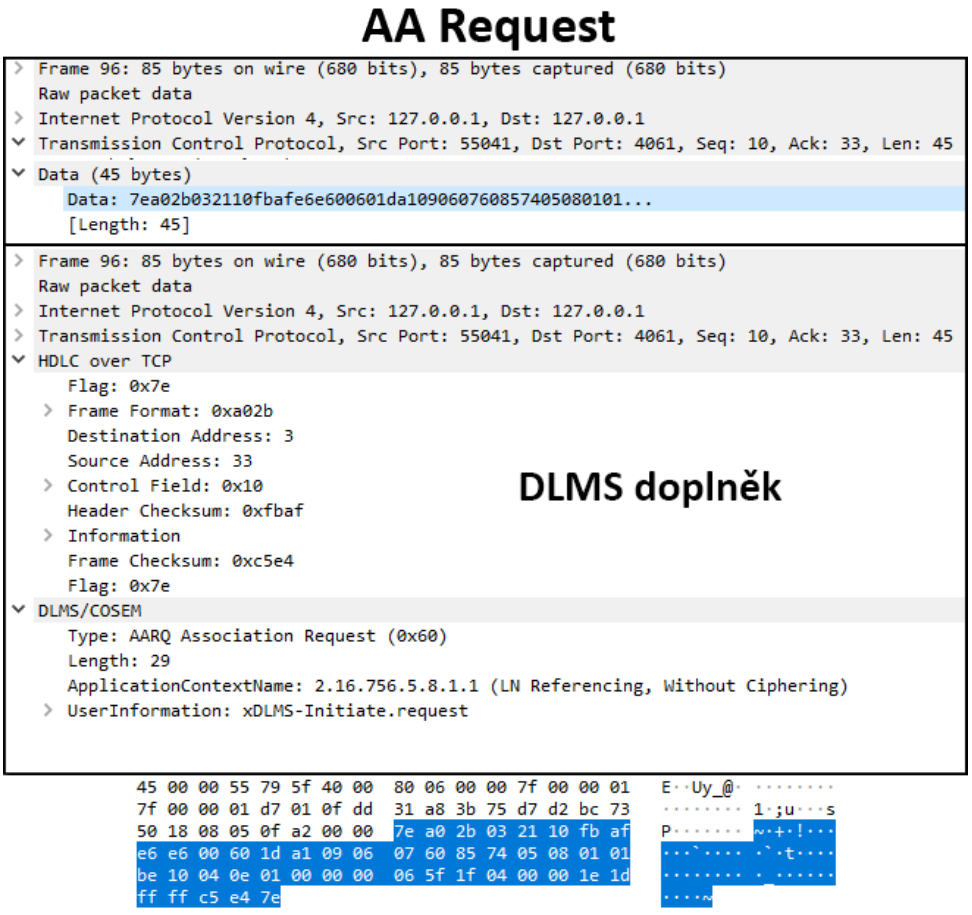
Kódy skupiny F
Poslední skupina může popisovat účtovací období. Obsahuje i částečnou historii. Může obsahovat různá schémata, například týdenní nebo měsíční. Každé schéma obsahuje čítač počtu období, počet volných období, časovou známku posledního vyúčtování včetně historie. Jako poslední se uvádí délka jednoho období. [8]

5 Analyzátor provozu

Pro analýzu provozu protokolu DLMS je použit program Wireshark [15] s doplňkem pro rozpoznání DLMS zpráv. Tento doplněk pochází od doktora z fakulty informačních technologií VUT, pana Ing. Petra Matouška, Ph.D. [16], který se v roce 2017 zabýval právě analýzou tohoto protokolu ve své technické zprávě [17].

Použitím programu Wireshark je možné analyzátor realizovat na různých platformách, včetně OS Linux. Bez tohoto doplňku jsou viditelné pouze TCP zprávy, které nesou data DLMS zpráv. Ukázka, jak zpráva vypadá s doplňkem i bez něj lze vidět na průběhu komunikace při ustanovení AA. Žádost o AA (AARQ) je na obrázku 5.1.

Pro demonstraci analýzy je využit v roli klienta program od finské společnosti Gurux, GXDLMSDirector[1], který umožňuje snadné a rychlé přidání serveru a následné připojení, včetně čtení a zápisu dat na server. Pro analýzu bylo využito simulátoru serveru pomocí kódu v programovacím jazyce Java, taktéž od firmy Gurux [18].



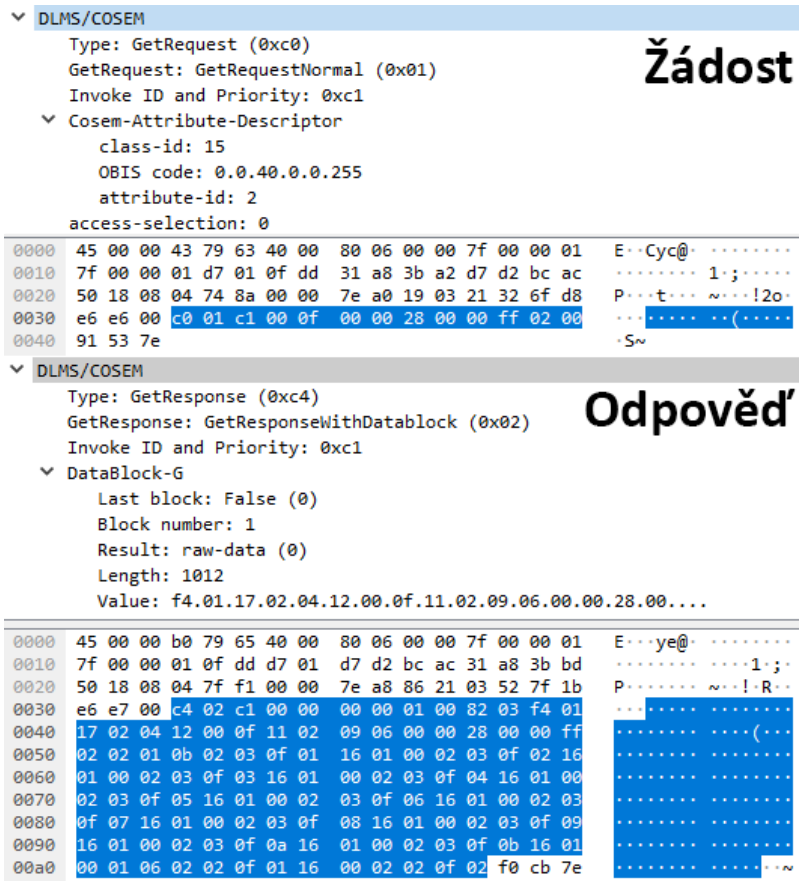
Obr. 5.1: Analýza DLMS při navazování AA – žádost

[1] http://gurux.fi/Download

Na obrázku 5.2 lze vidět první žádost po úspěšné AA. Zde se klient dotazuje serveru na asociaci jmen. Dotaz obsahuje OBIS kód 0.0.40.0.0.255, to je 6 hodnot, které odpovídají skupinám A–F. Jedná se tedy o abstraktní (A=0) žádost na asociaci jmen (C=40). V odpovědi je pouze část struktury se jmény. Na další části struktury klient zasílá další žádosti. Tyto zprávy lze také vidět na obrázku 5.3, kde řádky 100 a 102 reprezentují žádost a odpověď.

Dále si lze povšimnout, že po každé DLMS zprávě přichází i TCP potvrzení. Jedná se o normální chování stavového TCP protokolu, kde je každá zpráva potvrzována. První TCP zpráva je vlastně zakryta protokolem DLMS. Pro komunikaci lze také využít bezestavový protokol UDP, kde by potvrzovací zprávy nebyly zasílány.

Zprávy byly zachytávány mezi programem GXDLMSDirector a serverem spuštěným v programu Eclipse na počítači s OS Windows 10. Tato konfigurace ale vytváří problém se zachytáváním, jelikož oba programy běží v rámci jednoho OS. Program Wireshark však neumí zachytávat na lokálním rozhraní loopback. Proto byla komunikace odchytávána pomocí programu `RawCap` [19] jehož výstup byl následně zobrazen v programu Wireshark. Tato kombinace způsobuje, že komunikace tímto způsobem nelze zobrazovat v reálném čase.



Obr. 5.2: Žádost a odpověď na Asociaci SN/LN

No.	Time	Source	Destination	Protocol	Length	Info
96	30.181925	127.0.0.1	127.0.0.1	DLMS	85	DLMS AARQ Association Request
97	30.181925	127.0.0.1	127.0.0.1	TCP	40	4061 → 55041 [ACK] Seq=33 Ack=55 Win=525312 Len=0
98	30.181925	127.0.0.1	127.0.0.1	DLMS	97	DLMS AARE Association Response: accepted
99	30.181925	127.0.0.1	127.0.0.1	TCP	40	55041 → 4061 [ACK] Seq=55 Ack=90 Win=525312 Len=0
100	33.022838	127.0.0.1	127.0.0.1	DLMS	67	GetRequestNormal, class=15, OBIS=0.0.40.0.0.255, attr=2
101	33.022838	127.0.0.1	127.0.0.1	TCP	40	4061 → 55041 [ACK] Seq=90 Ack=82 Win=525312 Len=0
102	33.022838	127.0.0.1	127.0.0.1	DLMS	176	GetResponseWithDatablock no. 1, octet-string (112 bytes)
103	33.022838	127.0.0.1	127.0.0.1	TCP	40	55041 → 4061 [ACK] Seq=82 Ack=226 Win=525312 Len=0
136	33.044987	127.0.0.1	127.0.0.1	DLMS	61	GetRequestNext, block no: 1
137	33.044987	127.0.0.1	127.0.0.1	TCP	40	4061 → 55041 [ACK] Seq=1216 Ack=175 Win=525312 Len=0
138	33.044987	127.0.0.1	127.0.0.1	DLMS	176	GetResponseWithDatablock no. 2, octet-string (112 bytes)
139	33.044987	127.0.0.1	127.0.0.1	TCP	40	55041 → 4061 [ACK] Seq=175 Ack=1352 Win=524032 Len=0
160	33.123161	127.0.0.1	127.0.0.1	DLMS	67	GetRequestNormal, class=3, OBIS=1.1.21.25.0.255, attr=3
161	33.123161	127.0.0.1	127.0.0.1	TCP	40	4061 → 55041 [ACK] Seq=1924 Ack=247 Win=525312 Len=0
162	33.135791	127.0.0.1	127.0.0.1	DLMS	64	GetResponseNormal, structure (5 bytes)
163	33.135791	127.0.0.1	127.0.0.1	TCP	40	55041 → 4061 [ACK] Seq=247 Ack=1948 Win=525056 Len=0
164	33.136849	127.0.0.1	127.0.0.1	DLMS	67	GetRequestNormal, class=5, OBIS=1.0.31.4.0.255, attr=4
165	33.136849	127.0.0.1	127.0.0.1	TCP	40	4061 → 55041 [ACK] Seq=1948 Ack=274 Win=525312 Len=0
166	33.136849	127.0.0.1	127.0.0.1	DLMS	64	GetResponseNormal, structure (5 bytes)
167	33.136849	127.0.0.1	127.0.0.1	TCP	40	55041 → 4061 [ACK] Seq=274 Ack=1972 Win=525056 Len=0
168	33.144883	127.0.0.1	127.0.0.1	DLMS	67	GetRequestNormal, class=5, OBIS=1.0.31.4.0.255, attr=8
169	33.144883	127.0.0.1	127.0.0.1	TCP	40	4061 → 55041 [ACK] Seq=1972 Ack=301 Win=525056 Len=0
170	33.144883	127.0.0.1	127.0.0.1	DLMS	63	GetResponseNormal, double-long-unsigned (4 bytes)
171	33.144883	127.0.0.1	127.0.0.1	TCP	40	55041 → 4061 [ACK] Seq=301 Ack=1995 Win=525056 Len=0

Obr. 5.3: Průběh části komunikace mezi klientem a serverem

5.1 Překlad zpráv

Pro samotnou základní analýzu je pouhý Wireshark s doplňkem dostačující. Tato kombinace je vhodná pro rozpoznání DLMS zpráv a také umožňuje jednotlivé pakety třídit. Dále poskytuje základní informace o druhu zprávy, jako jsou například `getRequest`, `setRequest`, určení OBIS kódu nebo stavu (`accepted`, `rejected`, `other-reasons`) atp. Pro hlubší analýzu je vhodné navíc použít DLMS překladač, který je dostupný například na stránkách Gurux[2].

Během komunikace si díky doplňku vybereme DLMS zprávu. Jelikož zpráva obsahuje data celého paketu, je nutné vybrat pouhou DLMS část, kterou lze následně zkopírovat jako `Hex-Stream`.

Pro ukázku je zpráva z obrázku 5.2 zobrazena na výpisku 5.1. První dva řádky ukazují, jak vypadá zpráva jako hexadecimální string. Poté následuje výpis z překladače. Červeně je zobrazena pouze část s PDU, která se dá přeložit samostatně. V překladači je několik záložek. Právě červená část spadá do záložky „PDU“. V případě použití celé zprávy se využije záložka „Messages“, kde zpráva obsahuje i zapouzdření, které je v tomto případě HDLC. Při překladu celé zprávy se dozvíme i důležité hodnoty, zejména `TargetAddress` a `SourceAddress`, které udávají ID klienta a serveru. Překladač dále umí po zadání OBIS kódu říci, o jaký objekt se jedná z pohledu specifikace. Pokud má výrobce objekty vytvořené podle sebe špatně, je potom tato informace chybná.

[2]http://www.gurux.fi/GuruxDLMSTranslator

Na výpisku lze dále vidět, že se používá `HLS` autentizace. Způsob odkazování je pomocí `logical name` a žádá se zde o objekt `0.0.40.0.0.255`. Tento objekt běžně znamená aktuální asociaci objektů.

Jedna z posledních hodnot je zde `AttributeID`. Ten zařízení říká, která položka z objektu ho zajímá. V případě, že by tato hodnota byla 1, tak se jedná o samotné LN jméno. Ve vyobrazeném případě je tato hodnota 2 a jedná se zde o list všech objektů.

Server tedy zpátky zašle v několika zprávách všechna aktuální LN jména.

Výpis 5.1: Zobrazení zprávy za pomocí DLMS překladače

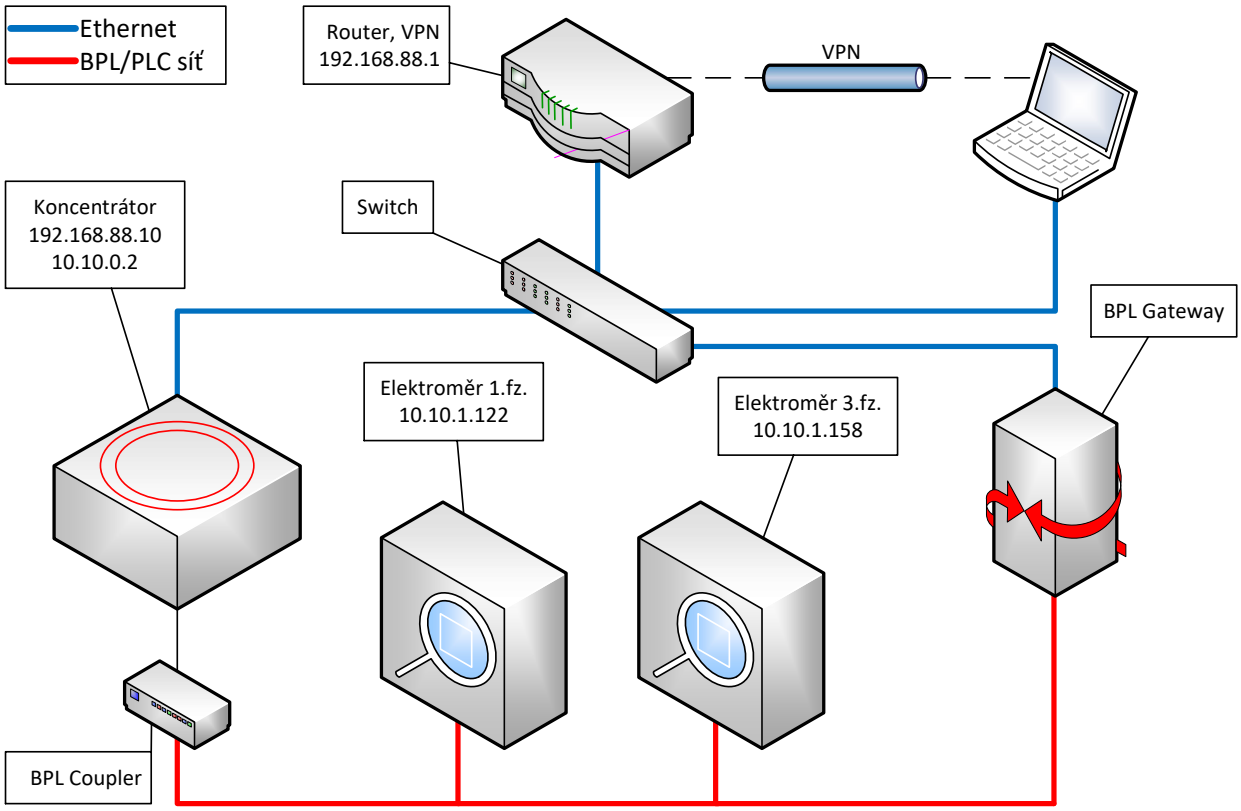
```
 7E A0 19 03 21 32 6F D8 E6 E6 00 C0 01 C1
 00 0F 00 00 28 00 00 FF 02 00 91 53 7E

<HDLC len="24" >
<TargetAddress Value="1" />
<SourceAddress Value="16" />
<!--I frame.-->
<FrameType Value="32" />
<PDU>
<GetRequest>
  <GetRequestNormal>
    <!--Priority: HIGH ServiceClass: CONFIRMED invokeID: 1-->
    <InvokeIdAndPriority Value="193" />
    <AttributeDescriptor>
      <!--ASSOCIATION_LOGICAL_NAME-->
      <ClassId Value="15" />
      <!--0.0.40.0.0.255-->
      <InstanceId Value="0000280000FF" />
      <AttributeId Value="2" />
    </AttributeDescriptor>
  </GetRequestNormal>
</GetRequest>
</PDU>
</HDLC>
```

6 Laboratorní zařízení

Tato kapitola popisuje laboratorní zařízení, jejich topologii a problémy se zařízeními, které byly zjištěny při realizaci generátoru.

6.1 Topologie

Generátor se může připojovat k elektroměru v laboratoři pomocí VPN, případně přímo do přítomného přepínače. Topologie rozložení zařízení v laboratoři je vyobrazena na obrázku 6.1. Zapojení začíná u směrovače, který také zajišťuje připojení přes VPN. Pro rozšíření počtu portů je směrovač propojen s přepínačem. Z neznámých důvodů funguje přítomný koncentrátor pouze s gigabitovými prvky. Nejspíše se předpokládá, že koncentrátor bude využíván naplno a bude tedy komunikovat s velkým množstvím elektroměrů a výsledná data budou větších objemů.



Obr. 6.1: Topologie zařízení v laboratoři

Prvním z prvků chytré sítě je již zmíněný koncentrátor, který má přidělenou adresu `192.168.88.10`. Při zadání této adresy do prohlížeče s využitím `HTTPS` a portu `8080` se lze připojit do webového rozhraní. Toto rozhraní umožňuje zobrazit přehled připojených elektroměrů a dále umožňuje čtení jednotlivých COSEM objektů.

Dále zde lze měnit nastavení samotného koncentrátoru. Koncentrátor je nastaven tak, že zajišťuje přemostění z BPL sítě do sítě Ethernet. Jedná se o adresy `10.10.x.x`. Koncentrátor bohužel sám o sobě neumožňuje připojení do BPL sítě a potřebuje k tomu malé zařízení – BPL Coupler, který se připojuje do koncentrátoru pomocí přítomného USB konektoru.

Nejdůležitější pro tuto práci jsou elektroměry. V laboratoři jsou přítomny dva, jeden třífázový a jeden jednofázový. Veškerá komunikace od elektroměrů je vedena po elektrických rozvodech (PLC), přesněji se jedná o přenos s využitím BPL. Díky tomu se nemusí k elektroměrům vést další např. Ethernetové kabely nebo používat přenos vzduchem. Další výhodou je, že spolu mohou jednotlivé elektroměry komunikovat a tak postupně přes sebe přenášet data i do větších vzdáleností. K následnému připojení více elektroměrů k řídícímu středisku lze využít koncentrátor nebo BPL bránu.

Posledním prvkem z BPL sítě je již zmíněná brána. Ta slouží k přemostění BPL a Ethernet sítě. Každý elektroměr pak dostane IP adresu od případného DHCP serveru v síti. Avšak pokud jsou elektroměry již připojené ke koncentrátoru, brána již nedokáže tyto zařízení připojit k routeru. Pokud se však koncentrátor odpojí/-vypne, elektroměry mohou získat adresu přes DHCP. Této skutečnosti by se dalo využít k případnému útoku na jednotlivé elektroměry a například číst spotřebu, či dokonce vypnout jednotlivá relé případně rozpojit přítomný stykač a tím zamezit napájení celého objektu. Stačí pak pouze vědět, kde se koncentrátor nachází a nějakým způsobem ho vyřadit z provozu. Následně připojit vlastní BPL bránu a router a tím získat přístup k elektroměrům.

Na obrázku 6.1 jsou vyobrazeny jak brána, tak i koncentrátor. V provozu je však vždy jen jedno zařízení a podle toho dostávají elektroměry adresy.

Seznam Smart Metering zařízení v laboratoři:

- Koncentrátor – SCU
- BPL Coupler
- Elektroměr jednofázový
- Elektromer třífázový
- BPL brána

6.2 Překážky u laboratorních zařízení

Specifikace DLMS/COSEM se snaží vést výrobce k tomu, aby mohly být vzájemně použity zařízení od různých výrobců. K tomu bohužel nespecifikuje určité části, díky kterým jsou různá zařízení vzájemně nekompatibilní.

Právě zařízení, které jsou přítomné v laboratoři nemají část specifikace implementovanou podle specifikace, ale podle výrobce. Z tohoto důvodu byla tvorba generátoru značně zpomalena. Nejprve bylo potřeba úspěšně navázat spojení mezi klientem ⇔ generátorem a serverem ⇔ elektroměrem.

Hlavním problémem byla neznalost nastavení, zejména druh použité autentizace, klientské a serverové ID, použitý druh odkazování, port, typ zapouzdření a v neposlední řadě i heslo pro autentizaci. Jediným známým parametrem byla IP adresa elektroměru, která lze při použití brány vyčíst z DHCP serveru, v případě koncentrátoru se adresa nachází v jeho seznamu zařízení. Port lze určit například pomocí skenování portů. Při špatné volbě portu zařízení odmítne TCP spojení, při správné volbě se TCP spojení vytvoří. V případě přítomných zařízení je použitý standardní port pro DLMS (`4059`).

Pokud je známa pouze IP adresa a port, zařízení pomocí DLMS neodpoví ani chybovou zprávou. Dalším potřebným a nezbytným údajem pro připojení je `clientID`, které lze zjistit více způsoby:

1. Generování AARQ zpráv a postupné zkoušení `clientID`
2. Odposlech komunikace mezi koncentrátorem a elektroměrem

První metoda je výhodná pro nalezení `clientID` v případě, kdy nemůžeme zachytit komunikaci. Při vyzkoušení této metody v laboratoři bylo zjištěno, že server nereaguje na ID od 1 do 15. Na dalších 10 zkoušených hodnot už z 90 % odpovídá. Správná hodnota u přítomných zařízení je 17. Pokud však použijeme první úspěšné tj. 16, tak i při dalším správném nastavení nebude spojení úspěšné.

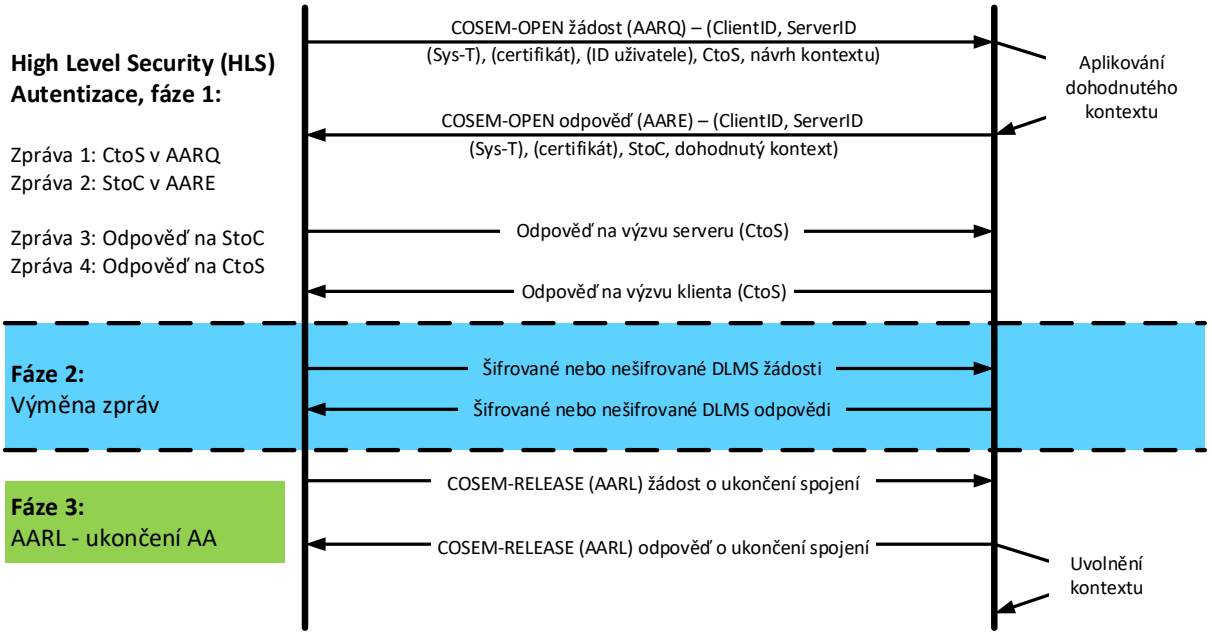
Druhá metoda mohla být provedena díky absenci šifrované DLMS komunikace a tak bylo možné všechny neznámé parametry zjistit ze zachycené komunikace mezi koncentrátorem a elektroměrem. Na nově vytvořenou AARQ zprávu, včetně všech nastavení, již server odpoví `accepted` a navíc zašle svoje heslo, které následně klient porovnává se svým záznamem. Pokud se shoduje, může následovat další krok. V následujícím kroku je potřeba dokončit autentizaci tzn. zaslat si vzájemné odpovědi na výzvy. To je zde provedeno uvnitř `ActionRequest` a `ActionResponse` zpráv. Průběh podle specifikace je vyobrazen na obrázku 6.2.

Opět se však ani tento krok neobešel bez překážek. Předpokládaný způsob výpočtů je v přítomných zařízeních implementován jinak. Při zaslání zpráv, vytvořených normálně podle specifikace, je spojení ukončeno chybovou hláškou `other-reasons`. Vyřešeno to bylo opět analýzou zachycené komunikace. Zároveň bylo zjištěno, že server vyžaduje specifickou výzvu ve zprávě AARQ. Tato výzva je statická a její hodnota byla také zachycena.

Zprávy Action Request a Response, jsou nositeli odpovědí na autentizace. Ve směru od klienta k serveru byla zpráva nahrazena zachycenou zprávou. Odpověď

serveru s výsledkem výzvy by u klienta neprošla, protože očekává odpověď na jinou výzvu. Kontrola této zprávy byla vyřešena změnou porovnávaných dat na očekávaná data. Tím v podstatě na straně klienta ignorujeme autentizaci serveru. Pro účely generátoru postačuje, aby se server domníval, že komunikuje s legitimním klientem.

Autentizace tedy byla překonána díky použití statických výzev a absenci šifrovaných zpráv. Principiálně se jedná o `reply attack`. Způsob, jakým je obcházení autentizace řešeno v generátoru, je popsán v kapitole 7.3.



Obr. 6.2: Průběh HLS autentizace [7]

Seznam zjištěných překážek:

1. **Port** – 4059
2. Způsob **zapouzdření** – wrapper
3. **ClientID** – 17
4. Druh autentizace – high
5. Způsob odkazování – LN
6. ServerID – 1
7. Přítomnost statické výzvy klienta k serveru
8. Hodnota výzvy
9. Heslo pro autentizaci serveru ke klientu
10. Druh výpočtu výzev

Znalost prvních třech bodů je nezbytná pro získání alespoň chybové odpovědi.

Další nalezené problémy na zařízeních jsou neimplementovaný asociační objekt `0.0.40.0.0.255`, místo toho se musí použít první asociace `0.0.40.0.1.255`. Poslední nalezenou chybou je špatný formát času, který je popsán v kapitole 6.2.1

6.2.1 Špatný formát času

Tato část velmi stručně popisuje standardní formát času používaný v DLMS a jeho špatnou implementaci na přítomných zařízeních [18].

Čas na elektroměru je přítomný pod objektem `0.0.1.0.0.255:2` a je ve formátu `OCTET-STRING`.

Odpověď serveru `GetResponse` s hodnotou času vypadá následovně: (Červeně je část obsahující čas) `0001000100110012c401c100090c07e3050f030d082bff008080`

Pro lepší zobrazení je tato hodnota rozdělena na výpisku 6.1. Hodnoty rok (YYYY), měsíc (MM), den (DD), hodina (HH), minuta (mm) a sekunda (ss), bylo snadné rozponat. Další hodnotou je den v týdnu (DW). Po sekundách je možnost použití i milisekund (ms), ale v tomto případě mají hodnotu FF, což znamená, že je tato hodnota přeskočena. Poslední hodnota je status hodin (st), kde `80` znamená, že je používán letní čas.

Problematickou hodnotou na přítomných zařízeních je položka, která udává časovou zónu (zone). Podle hodnoty 0080 by se jednalo o posun o 128 minut. Pokud by místo hodnoty `0080` byla nastavena hodnota `8000`, tak by se časová zóna ignorovala. Při nastavení času z aplikace a jeho následné vyčtení z meteru se díky špatné zóně zobrazí čas o 2 hodiny napřed. Tato skutečnost nastává dokonce i při nastavení času z webového rozhraní koncentrátoru. Pokud se však hodnota pouze synchronizuje, tak je zobrazena opět správně.

Výpis 6.1: Formát času v DLMS

YYYY	MM	DD	DW	HH	mm	ss	ms	zone	st
07e3	05	0f	03	0d	08	2b	ff	0080	80
2019	05	15	st	13	08	43			

Další možné hodnoty stavu hodin (poslední položka `st`) [18]:

- OK(0)
- INVALID_VALUE(0x1)
- DOUBTFUL_VALUE(0x2)
- DIFFERENT_CLOCK_BASE(0X4)
- INVALID_CLOCK_STATUS(0x8)
- RESERVED2(0x10)
- RESERVED3(0x20)
- RESERVED4(0x40)
- DAYLIGHT_SAVE_ACTIVE(0x80)
- SKIPPED(0xFF)

7 Generátor DLMS zpráv

Tato kapitola obsahuje analýzu vytvořeného generátoru. Rozebírá jeho důležité prvky a popisuje části zdrojového kódu. Zobrazený zdrojový kód není 100% shodný. Pro účely jeho rozboru byly některé části upraveny, případně odstraněny. Číslo řádku zdrojového kódu, který je v popisován, je uvedeno v závorkách ⟨⟨číslo⟩⟩.

7.1 Využité prostředky pro generátor

Vzhledem k tomu, že specifikace je velmi rozsáhlá a obsahuje velké množství objektů, tak je pro generátor využita knihovna DLMS od firmy Gurux [18], která je šířena pod duální licencí [20]. V tomto případě se jedná o open source a vztahuje se na něj licence GNU GPL v2 [21]. Společnost Gurux má velké zkušenosti s protokolem DLMS, kterému se věnuje již od roku 1998. Také má aktivní uživatele poskytující pomoc na fóru.

Generátor je realizován v jazyce Java s využitím vývojového prostředí Eclipse na počítači s OS Windows 10. Pro usnadnění zadávání a lepšího uživatelského dojmu z generátoru je vytvořeno grafické rozhraní (GUI) pomocí knihoven JavaFX.

Díky využití programovacího jazyka Java je generátor přenositelný mezi platformami a nemusí být tak vázán na konkrétní počítač či architekturu.

Seznam použitých knihoven a nástrojů:

1. Vývojové prostředí Eclipse IDE 2018-12 [22]
2. Kompilátor Java JDK-1.8.0_191 [23]
3. Nástroj Maven 4.0.0 [1]
4. Knihovna gurux.net – 1.0.16
5. Knihovna gurux.dlms – 2.2.26 [18]
6. Knihovna javafx – 8.8.3
7. Program JavaFX SceneBuilder – 8.5.0 [24]

7.2 Uživatelské rozhraní

Uživatelské rozhraní (GUI) bylo vytvořeno pomocí programu SceneBuilder a knihoven JavaFX. V této části je grafické rozhraní generátoru blíže popsáno. Při najetí myší na většinu tlačítek a textových polí se zobrazí malá nápověda, která popisuje k čemu daný prvek slouží, případně co by měl obsahovat.

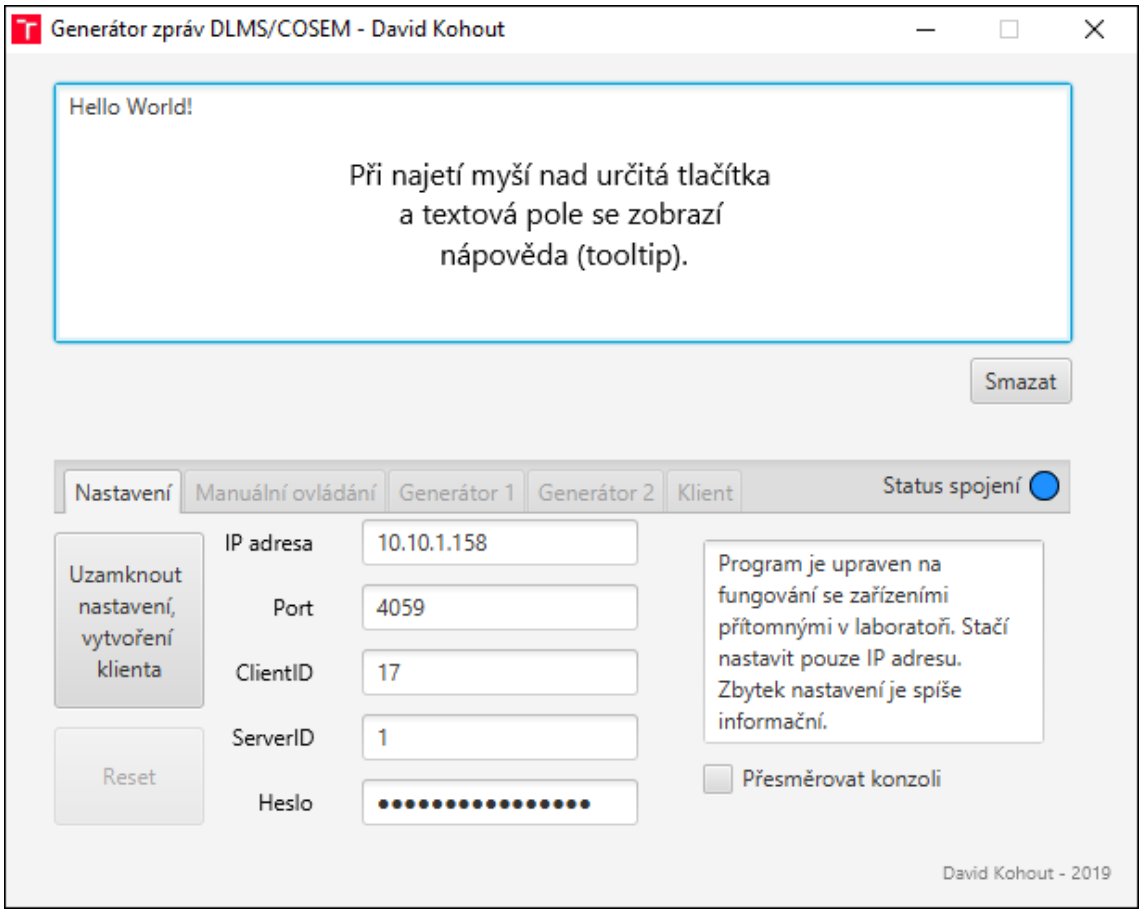
7.2.1 Úvod, nastavení

Po spuštění programu se otevře okno generátoru, které lze vidět na obrázku 7.1. Aplikace se dělí na dvě části. Spodní část je přepínatelná pomocí záložek a horní část je pro všechny tyto záložky stejná. V horní části se nachází textové pole, které

[1] Distribuováno s Eclipse a knihovny jsou dostupné přes Maven jako „dependencies“

slouží jako výstup programu. Nejdůležitější zprávy o chodu programu se zde vypisují. V podstatě se jedná o sekundární konzoli. Vpravo pod touto konzolí se nachází tlačítko, které ji umožňuje vymazat. Uprostřed pod konzolí je umístěno další textové pole, které se zobrazuje pouze při běhu delších operací a zobrazuje jejich průběh. Představuje především počítadlo zaslaných zpráv. Posledním prvkem, který je společný pro všechny záložky je indikátor stavu spojení, který mění barvu podle stavu. Aktualizace indikátoru je prováděna každých 5 vteřin.

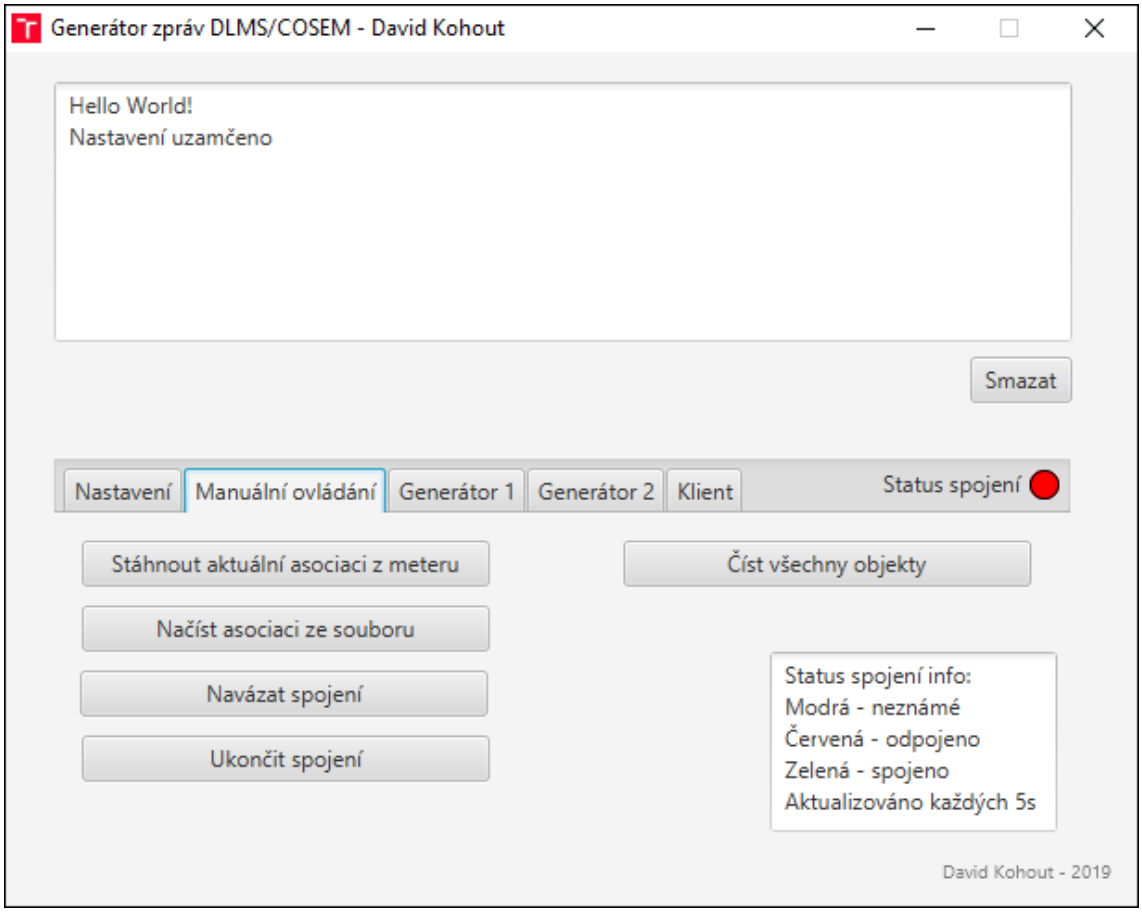
Dolní část v tomto případě obsahuje základní nastavení klienta (IP adresu, port, klientské ID, serverové ID a autentizační heslo). Další nastavení jako je například druh odkazování, typ zapouzdření a stupeň autentizace je natvrdo nastavené přímo v kódu. Bez správného a následného uzamčení nastavení, aplikace neumožní další ovládání. V případě potřeby navázání spojení na jinou IP adresu lze nastavení resetovat a změnit jakékoliv pole podle požadavků. Nakonec tato záložka ještě umožňuje zatrhnutí možnosti, která přesměruje veškerý výstup konzole do textového pole v aplikaci. Bohužel je toto přesměrování spíše experimentální a je značně problematické, proto není jeho použití doporučeno. Navíc většina výstupů je vypisována 2x, takže při zaškrtnutí by se zde objevovaly duplicitní zprávy.



Obr. 7.1: Úvodní obrazovka programu

7.2.2 Manuální ovládání

Tato část aplikace je vyobrazena na obrázku 7.2. První tlačítko `Stáhnout aktuální asociaci z meteru` naváže spojení s elektroměrem a dotazuje se na asociační objekt. Normálně má tento objekt LN jméno `0.0.40.0.0.255` a znamená aktuální asociaci objektů na meteru. Bohužel smart metery v laboratoři nemají tento objekt implementovaný a muselo být použito dalšího objektu, který také obsahuje asociaci. Jedná se o objekt `0.0.40.0.1.255` a běžně mají mít zařízení oba objekty, i když se jedná v podstatě o to samé. Po připojení a stažení asociace se uloží seznam jmen do souboru. Druhé tlačítko využívá tento soubor k načtení asociace bez připojování k elektroměru.



Obr. 7.2: Manuální ovládání

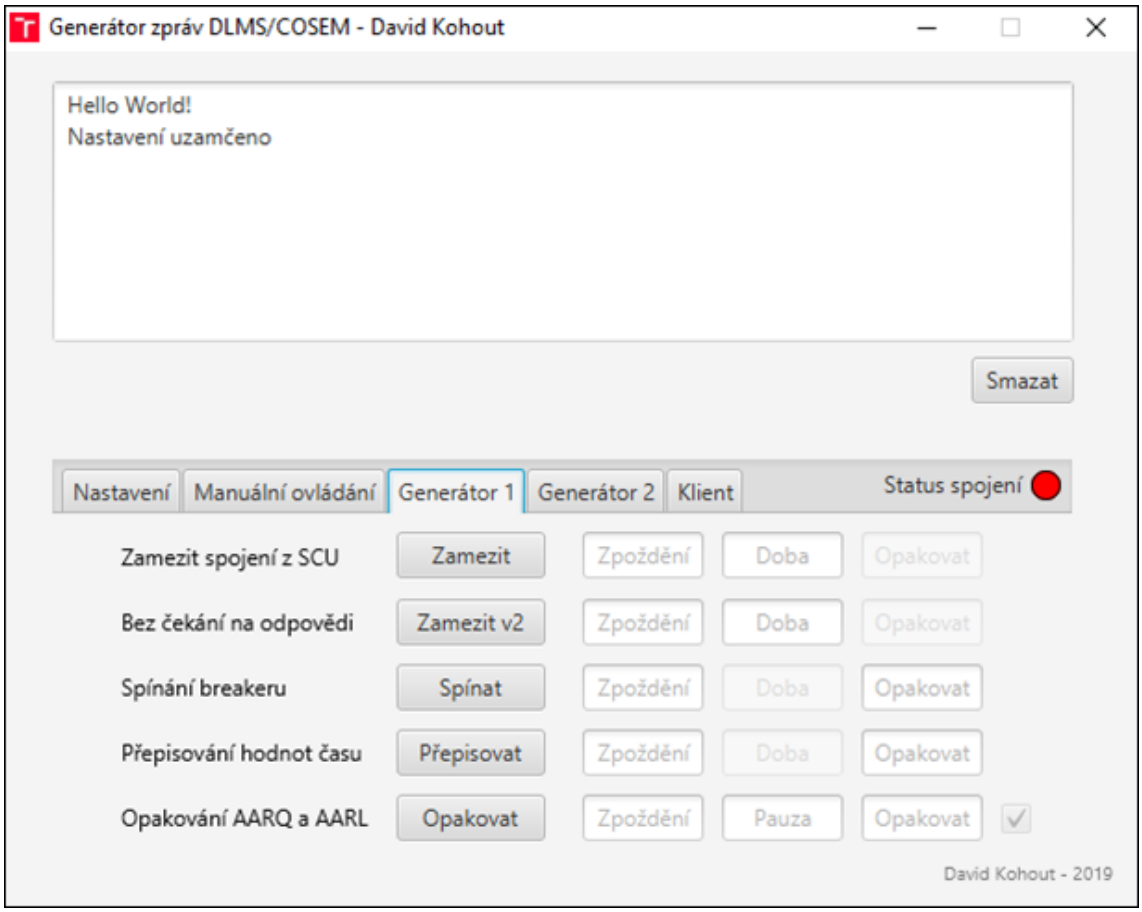
Dále jsou zde tlačítka pro manuální navázání spojení, které zasílají AARQ a následně ActionRequest zprávy. Další tlačítko ukončuje toto spojení zasláním zprávy AARL a následně ukončí i TCP spojení.

Posledním tlačítkem v této části je čtení všech objektů. Po stisknutí se naváže spojení, stáhne se asociace a následně se začnou číst všechny objekty i jejich atributy. V textovém poli se čtené objekty začnou vypisovat nejdříve minutu po stisknutí

tlačítka. Je to způsobeno právě čtením asociace a zjišťováním vlastností objektů implementovaných výrobcem. Celková doba čtení všech objektů může trvat i 10 minut.

7.2.3 Generátor – záložka 1

Do aplikace bylo vytvořeno několik typů generování zasílaných zpráv. Pro přehlednější zobrazení proto byly rozděleny do dvou záložek v aplikaci. První lze vidět na obrázku 7.3. Některé textové pole jsou vypnuté, protože nakonec nebyly při generování použity. Všechny jinak obsahují nápovědu pro správné nastavení. Při jejich chybném nebo úplném nevyplnění se použijí výchozí hodnoty.



Obr. 7.3: Generátor – záložka 1

První dvě tlačítka mají stejnou funkčnost, pouze se u druhé nečeká na přijetí odpovědi. V jejich generátorech se postupně prochází seznam objektů a dotazují se na první atribut, který obsahuje samotné LN jméno.

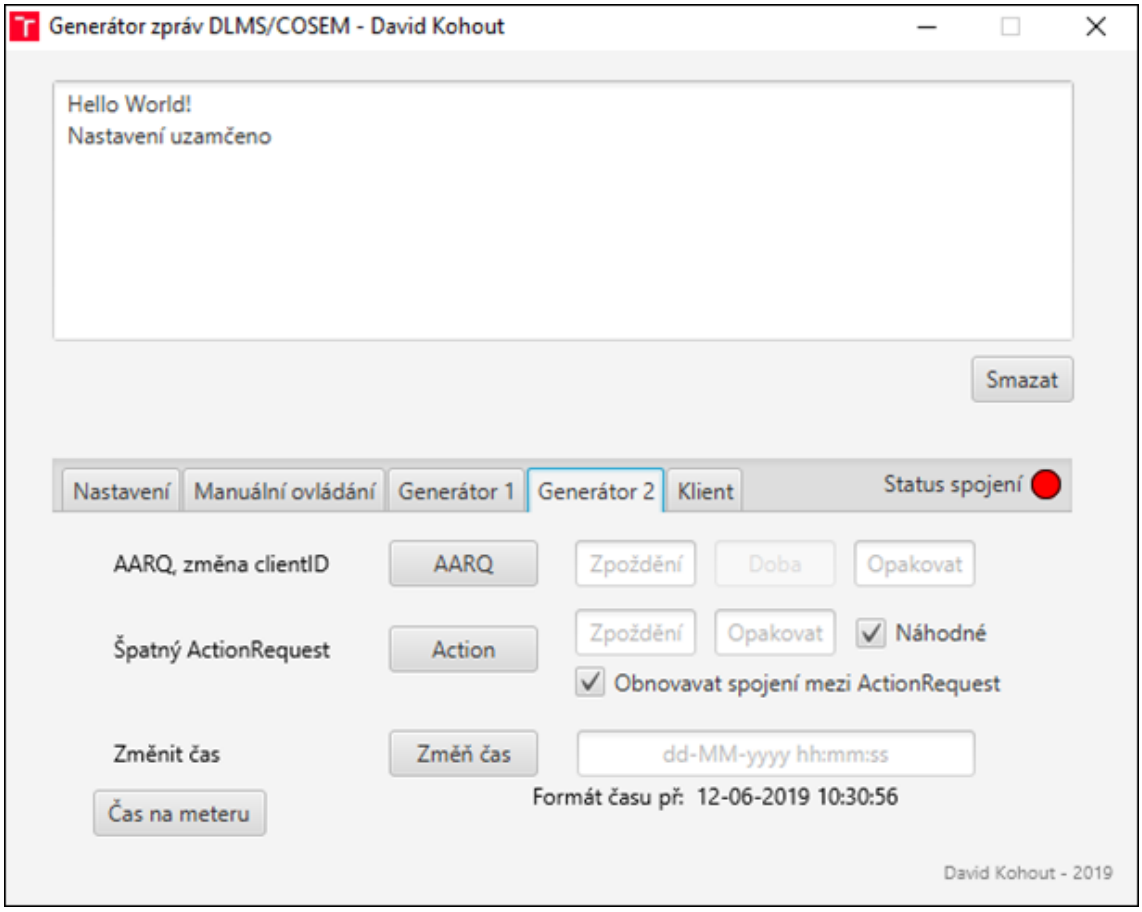
Tlačítko `Spínat` dokáže po zapnutí opakovaně spojovat a odpojovat přítomné relé v elektroměru, které může sloužit k odpojení neplatičů od elektrické energie bez nutnosti manuálního odpojení přívodu.

Další funkce je přepisování času. Čas se při každé zaslané zprávě změní pouze v jedné hodnotě (den, měsíc, rok, hodina, minuta nebo sekunda).

Poslední položka opakovaně navazuje spojení (AARQ, ActionRequest a AARL).

7.2.4 Generátor – záložka 2

Na druhé záložce generátoru z obrázku 7.4 se nachází dvě další možnosti generování a část pro manuální změnu času. Obě tyto možnosti generování nepřistupují na meter legitimně. První generuje AARQ zprávy a postupně mění hodnotu `clientID` od 1 do hodnoty podle položky opakovat. Při sledování komunikace nebude zařízení odpovídat, dokud nebude ID správné.



Obr. 7.4: Generátor – záložka 2

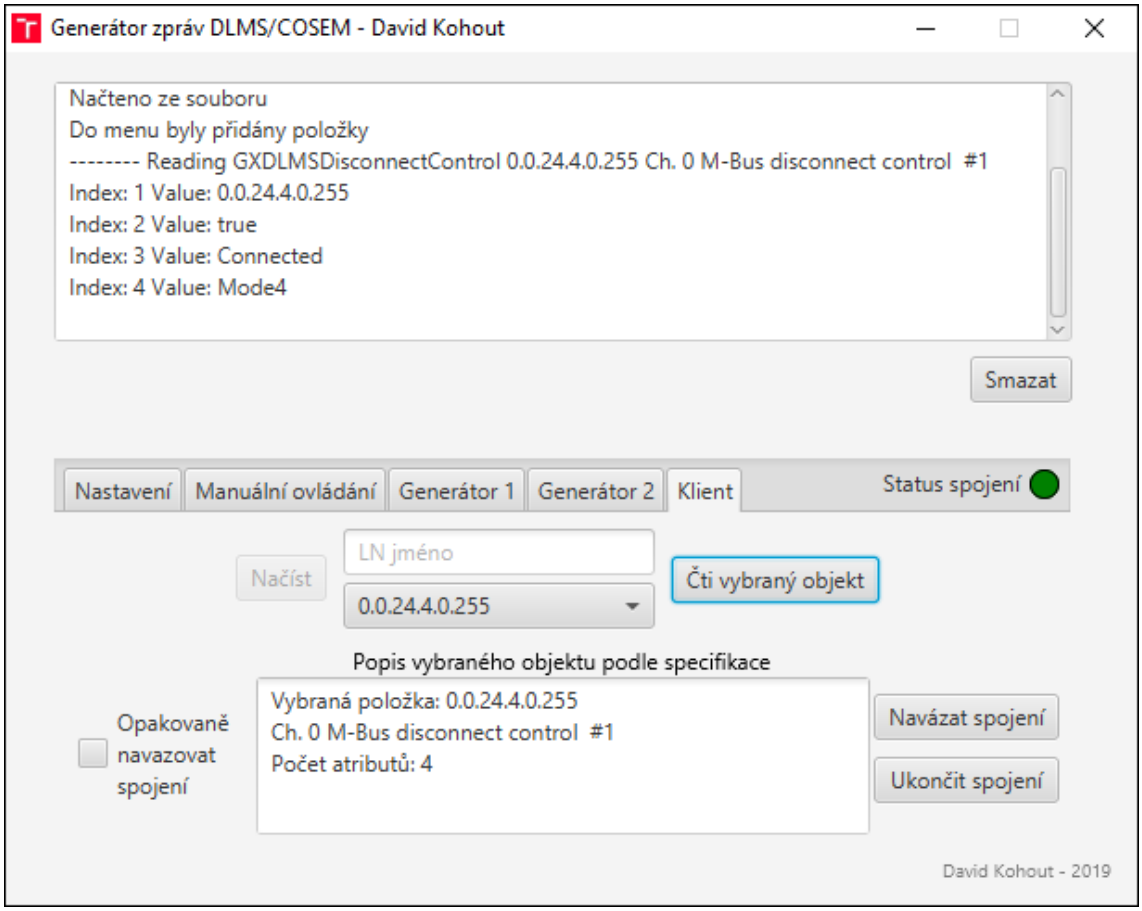
Druhé tlačítko již má správné `clientID`, ale nezná správný postup výpočtu odpovědi na výzvu. Takže se zpráva ActionRequest zasílá s úplně náhodnou odpovědí (při zatržení možnosti náhodné) nebo odpovídá podle výpočtu, který vychází ze specifikace. Položku `Obnovovat spojení mezi ActionRequest` je doporučeno nechat zatrženou, protože v rámci jednoho spojení může být zaslána pouze jedna odpověď na výzvu. V případě zaslání více zpráv server odpovídá `ServiceUnsupported`.

Poslední částí na této záložce je práce s časem. Pokud se do přítomného pole nastaví čas podle odpovídajícího formátu, tak dojde ke změně času na elektroměru. Bohužel takto nastavený čas nezůstane dlouho, jelikož ho koncentrátor velmi často kontroluje a v případě neshody jej změní, aby odpovídal vlastním hodinám.

Pro kontrolu času se zde nachází tlačítko pro jeho zjištění. Právě při tvoření této funkce byla zjištěna nesrovnalost s formátem času na meteru, více v kapitole 6.2.1

7.2.5 Klient

Tato část aplikace, která je zobrazena na obrázku 7.5, umožňuje manuální čtení vybraných objektů. Prvně je zde potřeba stisknout tlačítko `Načíst` pro načtení všech objektů do výběrového pole. Poté je možné si vybrat požadovaný objekt buď z menu, nebo napsat jeho LN jméno do určeného pole. Pokud by se zadané jméno nenacházelo v menu, nedojde k jeho čtení. V případě, že je zadaná hodnota správná, tak se automaticky vybere položka v menu a dojde k jeho následnému čtení.



Obr. 7.5: Klient – manuální čtení objektů

Ve velkém textovém poli v dolní části se vypisuje popis objektu a počet jeho atributů. Tyto informace však pochází ze specifikace a mohou se lišit. Občas může

dojít i ke čtení atributu, který se již na zařízení nenachází a poté se vytiskne krátká chyba do výpisu. Při výběru jména z menu se automaticky zobrazí popisek objektu. V případě zadání jména ručně se teto popisek zobrazí až po stisknutí tlačítka čtení.

Nakonec se zde opět nachází tlačítka pro manuální navázání a ukončení spojení. Společně z možností zatrhnutí `Opakovaně navazovat spojení` mohou pomoci k rychlejšímu čtení více objektů. Pokud je políčko nezatržené, tak se spojení naváže automaticky po stisknutí čtení. Poté se spojení samo neukončí. Případně až po uplynutí cca 60 sekund, kdy dochází k automatickému ukončení TCP spojení.

7.3 Vyřešení problému s autentizací

Autentizace je nezbytná pro navázání spojení mezi elektroměrem a generátorem. Jak bylo popsáno v kapitole 6.2, tak s autentizací u dostupných zařízení bylo nejvíce problémů. Tato část popisuje, jak generátor obchází tyto problémy. Na jejich vyřešení bylo potřeba upravit třídy z knihoven Gurux. Tyto třídy začínají na písmeno A. Lze je tak snadněji rozeznat od jejich původních tříd.

Na výpisku 7.1 je zobrazena metoda z třídy `AGXDLMSReader`. Tato třída má několik částí upravených nebo přidaných pro lepší fungování s vytvořenou aplikací. Zobrazená metoda je jednou z nejdůležitějších metod – `initializeConnection()`. Tato metoda se stará o navázání DLMS spojení. Zasílá tedy zprávy AARQ ⟨⟨4⟩⟩ a ActionRequest ⟨⟨25⟩⟩. Právě druhá zmíněná obsahuje odpověď na výzvu od serveru. Elektroměr tuto část očekává pouze v jediné podobě. Ve výpisku je tato část ve `Stringu hexString` ⟨⟨11⟩⟩. Konkrétně se jedná o jeho čtvrtou část ⟨⟨14⟩⟩, která je 32 znaků dlouhá. Z bezpečnostních důvodů je tato část tvořena pouze znaky A.

Celá zpráva se získá příkazem `dlms.getApplicationAssociationRequest()` ⟨⟨9⟩⟩. Tím se ale získá zpráva s jinou než potřebnou odpovědí. Proto byla do kódu přidána část, která převádí hexadecimální string do bytového pole ⟨⟨17-23⟩⟩. Toto pole je následně zasláno na server pomocí `readDLMSPacket` ⟨⟨25⟩⟩. Objekt pojmenovaný jako `dlms` je objekt upravené třídy `AGXDLMSClient`.

Dále na začátku metody je definován objekt `GXDLMSReply` ⟨⟨2⟩⟩, do kterého se zapisují odpovědi, které jsou jednotlivě analyzovány ve třídě `AGXDLMSClient` v metodách `parseAareResponse` ⟨⟨6⟩⟩ a `parseApplicationAssociationResponse` ⟨⟨27⟩⟩. Tyto metody nastavují dodatečné parametry klienta a řeší výpis případných chyb. Pokud by kód neobsahoval přidanou část, tak aplikace končí chybou `other-reason`, která přijde z metody na konci ⟨⟨27⟩⟩.

Výpis 7.1: Vyřešení autentizace

```
00 void initializeConnection() throws Exception, InterruptedException {
01
02     GXReplyData reply = new GXReplyData();  //Objekt pro uložení odpovědi
03
04     readDataBlock(dlms.aarqRequest(), reply);   //Zaslání AARQ
05
06     dlms.parseAareResponse(reply.getData());    //Zpracování odpovědi
07     reply.clear();
08
09     for (byte[] it : dlms.getApplicationAssociationRequest()) { //Získání
10                                                                 //ActionRequest
11         String hexString = "000100110001001FC301C1000F"+//Hlavička DLMS zprávy
12                                                "0000280000FF"+//LN jméno objektu
13                                                    "01010910"+//Další část zprávy
14                         "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA";//Část s odpovědí
15                                                              //na výzvu
16
17         byte[] val = new byte[hexString.length() / 2];
18         for (int i = 0; i < val.length; i++) {
19            int index = i * 2;
20            int j=Integer.parseInt(hexString.substring(index, index + 2), 16);
21            val[i] = (byte) j;
22         }   //Převedení Stringu na byte[]
23         it = val.clone();           //Změna zprávy na upravenou
24
25         readDLMSPacket(it, reply);  //Zaslání ActionRequest
26     }
27     dlms.parseApplicationAssociationResponse(reply.getData());  //Zpracování
28                                                                   //odpovědi
29 }
```

Druhá část vyřešení problému autentizace je řešena podobným způsobem. Uvnitř metody `parseApplicationAssociationResponse` ⟨⟨27⟩⟩ dochází k porovnání získané hodnoty (odpověď serveru na výzvu) s hodnotou předpokládanou. Předpokládaná hodnota byla do kódu umístěna stejným způsobem. Jedná se o hexadecimální string o délce 32 znaků, který je převeden do bytového pole a tím nahrazuje předpokládanou hodnotu, která je následně porovnávána.

Další nutná oprava byla změna objektu pro asociaci. V průběhu běhu aplikace se musí zjistit seznam všech objektů na meteru. Při zjišťování se zavolá metoda, obsažena v třídě `AGXDLMSClient`, s názvem `getObjectsRequest`, ve které je nastavené správné LN jméno pro asociaci.

7.4 Druhy generovaných zpráv

Generátor obsahuje několik způsobů generování zpráv. Přesněji se jedná o 7 možností, které jsou v této části ve zkratce popsány. Všechny metody generování zpráv se nacházejí ve třídě `Generator`. Zdrojový kód každé této metody má podobný základ, který je zobrazený na obecném výpisu 7.2.

Výpis 7.2: Obecná metoda generování zpráv

```
00 public static void obecnaMetoda(NastavKlienta klient, int opakuj, int delay){
01     try {
02         Controller.counterVisible(true);    //zobrazení stavového řádku v app
03 
04         AGXDLMSReader reader = klient.getReader(); //získání
05                                                    //nastaveného readeru
06 
07         reader.Media.open();            //otevření TCP spojení
08         reader.initializeConnection();  //navázání DLMS spojení
09 
10         int pocet = 0;
11         for (int i=1; i<=opakuj; i++) {
12                 //cyklus, ve kterém se budou zprávy zpracovávat a odesílat
13 
14             //část metod umožňující zasílání zpráv
15             reader.read(objekt, attributeIndex);
16             reader.Media.send(data, receiver);
17             reader.readFast(data);
18             reader.readDataBlock(data, reply);
19 
20 
21          TimeUnit.MILLISECONDS.sleep(delay); //čekání mezi zasláním
22          pocet = i;                                          //další zprávy
23         }
24 
25         //výpis do konzole a do aplikace. Schování stavového pole
26         String w = "Počet zaslaných zpráv: "+pocet;
27         Controller.counterVisible(false);
28         Controller.pp(w+"\n");
29         System.out.println(w);
30 
31         reader.close();                 //ukončení spojení
32 
33     } catch (Exception e) {             //zachycení případné chyby
34         e.printStackTrace();
35         Controller.pp("Nastala chyba během funkce\n");
36     }
37 }
```

Všechny metody obsahují blok `try-catch` ⟨⟨11, 33⟩⟩ pro řešení případných chyb, týkajících se především spojení. Ke konci metody se vytiskne konečný stav a aplikace se uvede do předchozího stavu. V případě potřeby se zobrazí v aplikaci objekt zavoláním příkazu `Controller.counterVisible` ⟨⟨2, 27⟩⟩, ve kterém se vypisuje stav během generování a zasílání zpráv.

Další společnou částí je přiřazení nastaveného readeru ⟨⟨4⟩⟩, který obsahuje všechna nastavení klienta i spojení, na jednoduchou proměnou, která usnadňuje následné volání. Poslední společnou částí je přítomnost cyklu ⟨⟨11-23⟩⟩. Zbytek se již může velmi měnit.

Otevření ⟨⟨7⟩⟩ a ukončení ⟨⟨31⟩⟩ spojení obsahuje taky každý způsob, ale mohou se lišit umístění těchto volání a to buď uvnitř, nebo vně cyklu. Na výpisu se uvnitř cyklu nachází několik způsobů zaslání zpráv ⟨⟨15-18⟩⟩. Většina těchto metod ve výsledku volá jedna druhou, až skončí u druhé uvedené (`Media.send` ⟨⟨16⟩⟩), která se stará o samotné zaslání zpráv a řídí si přechod do nižších vrstev podle komunikačního modelu, blíže popsaného v kapitole 3.1. V cyklu se dále nachází zpoždění ⟨⟨21⟩⟩, které se dá nastavit zapsáním hodnoty do příslušného políčka při spouštění vybraného způsobu generování.

7.4.1 Zamezit spojení z SCU

Tato metoda generování zpráv je jako jediná v této práci doplněna i o kompletní výpis zdrojového kódu 7.3. Jedná se v podstatě o nejjednodušší metodu, která však bohatě stačí k zamezení připojení dalších zařízení na elektroměr.

Po spuštění metody dojde k úspěšnému navázání TCP ⟨⟨5⟩⟩ i DLMS ⟨⟨6⟩⟩ spojení. Následně se vyčte uložený seznam objektů ⟨⟨8⟩⟩ do nové kolekce označené jako `objekty`. Aby se při každém spuštění nevyčítaly objekty ve stejném pořadí, tak je kolekce náhodně zamíchána ⟨⟨10⟩⟩. Pomocné proměnné ⟨⟨11-13⟩⟩ slouží k počítání počtu průchodů a také pomáhají resetovat index objektů ⟨⟨19⟩⟩, při dosažení posledního objektu. V případě obou v laboratoři dostupných elektroměrů je počet shodný a to 438 položek. Při resetování indexu se kolekce opět zamíchá ⟨⟨20⟩⟩.

Pro generování je zde použit cyklus for ⟨⟨16⟩⟩, který kontroluje čas a při překonání zadané doby generování se ukončí. Následně pomocí `reader.read` ⟨⟨22⟩⟩ se vyšle žádost na určený objekt a jeho atribut 1. Právě atribut 1 bude obsahovat každý objekt, protože se jedná o samotné LN jméno, kterým je daný objekt volán. Nakonec se v cyklu vypíše status do aplikace ⟨⟨24⟩⟩, dojde k inkrementaci proměnných a vyčká se na uplynutí nastaveného zpoždění ⟨⟨27⟩⟩ před zasláním další zprávy.

Nakonec metody se uzavře spojení ⟨⟨36⟩⟩ a vypíší se patřičné informace do konzole i aplikace ⟨⟨31-34⟩⟩.

Výpis 7.3: Metoda zamezení spojení z SCU

```
00 public static void zamezitPristupSCU(NastavKlienta klient, int pocetSekund,
01                                                    int delayMiliMeziCtenim) {
02     try {
03         AGXDLMSReader reader = klient.getReader(); //získání
04                                                          //nastaveného readeru
05         reader.Media.open();            //otevření TCP spojení
06         reader.initializeConnection();  //navázání DLMS spojení
07 
08         GXDLMSObjectCollection objekty = reader.dlms.getObjects(); //získání
09                                             //seznamu všech objektů na meteru
10         Collections.shuffle(objekty);       //Náhodná změna pořadí objektů
11         int i = 1;
12         int pocetObjektu = objekty.size();  //pomocné proměnné
13         int indexObjektu = 0;
14         Controller.counterVisible(true);    //zobrazení stavového řádku v app
15 
16         for (long s=System.nanoTime()+TimeUnit.SECONDS.toNanos(pocetSekund);
17                                                      s>System.nanoTime();) {
18             if (indexObjektu == pocetObjektu) { //pro případ, kdy je cyklus
19                 indexObjektu = 0;               //delší než je počet objektů
20                 Collections.shuffle(objekty);
21             }
22             reader.read(objekty.get(indexObjektu), 1);  //zaslání zprávy
23 
24             Controller.counterPrint("Počet odeslaných zpráv: "+i);
25                                         //print stavu do aplikace
26             indexObjektu++; i++;
27             TimeUnit.MILLISECONDS.sleep(delayMiliMeziCtenim);
28                                         //čekání mezi zasláním další zprávy
29         }
30         //výpis do konzole a do aplikace. Schování stavového pole
31         String w = "Zamezení dokončeno, zasláno zpráv: "+i;
32         Controller.counterVisible(false);
33         Controller.pp(w+"\n");
34         System.out.println(w);
35 
36         reader.close();                 //ukončení spojení
37 
38     } catch (Exception e) {             //zachycení případné chyby
39         e.printStackTrace();
40         Controller.pp("Nastala chyba v zamezení spojení\n");
41     }
42 }
```

7.4.2 Bez čekání na odpovědi

Tato metoda je identická s předchozí, ale využívá jiný příkaz pro zaslání zprávy. Jedná se o `reader.Media.send`, jehož parametry jsou zpráva (bytové pole) a objekt pro zapsání odpovědi. Jelikož zde nečekáme na odpověď, tak je tento parametr nastaven na `null`.

Samotný tvar zprávy se získá pomocí `reader.dlms.read`, která má jako vstupní parametry objekt a atribut. Objekt pochází z kolekce a atribut je opět 1.

7.4.3 Spínání breakeru

Tato metoda spíná v elektroměru umístěné relé, které je označované jako „breaker“. Pomocí vyčtení všech přítomných objektů na elektroměru (log se nachází na CD) funkcí `ReadAll`, bylo zjištěno, že určený objekt odpovídá specifikaci a jeho jméno je `0.0.24.4.0.255` tzv. `DISCONNECT_CONTROL`. Atribut 2 odpovídá jeho stavu `true` nebo `false`. Bohužel použití zprávy typu `SetRequest` v tomto případě končí chybovou hláškou `ReadWriteDenied`.

Běžně by se tato zpráva získala pomocí `reade.dlm.write(objekt, atribut)`. Jelikož se však jedná o celkem kritický objekt, kterým lze odpojit celé odběrné místo, tak se musí nastavovací zprávy zasílat uvnitř `ActionRequest` zpráv. Takovou zprávu by šlo určitě vytvořit pomocí nějaké metody z třídy `AGXDLMSClient`, ale pro úsporu času byl tvar zprávy zachycen na koncentrátoru. Převod zprávy ze stringu do bytového pole je identický s popsaným postupem v kapitole 7.3.

Takto vytvořené zprávy jsou potom zasílané na elektroměr. Vždy se nejprve relé rozepne a poté sepne. Mezi každou změnou je aplikované nastavené zpoždění. V rámci jednoho cyklu se provedou obě operace.

Pokud je čas zpoždění příliš nízký, tak se změna nemusí na elektroměru vůbec projevit. Při přepnutí je slyšet cvaknutí relé. Čas mezi cvaknutími nemusí odpovídat času nastaveného zpoždění. To je způsobeno zpožděním při přenosu a zpracování zpráv a dále přítomné relé neumí rychle měnit stav, protože je konstruované především na stálé sepnutí.

7.4.4 Přepisování hodnot času

Tato metoda využívá zpráv typu `SetRequest` ke změně data a času na elektroměru. Jelikož se jedná o jiný typ generování, tak je úryvek kódu zobrazen na výpisu 7.4. Metoda po navázání spojení vstoupí do cyklu, kde se při každém průchodu změní jeden prvek času.

Nastavování času je zde řešeno pomocí objektu `Calendar` $\langle\langle 3 \rangle\rangle$, který umožňuje snadno měnit jednotlivé prvky data $\langle\langle 10 \rangle\rangle$. Který prvek se bude měnit určuje počet průchodů cyklem. Čas má 6 položek a tak se každá změní jednou za šest průchodů. Tvar zprávy je vytvořen pomocí `reader.dlms.write(...)` $\langle\langle 15 \rangle\rangle$. V tomto konkrétním případě je zde použit jiný způsob, než předání již hotového `GXDLMSObjectu`. Každá potřebná část je nastavena manuálně. Blíže je metoda popsána pomocí komentářů na výpisu.

Výpis 7.4: Část metody pro náhodnou změnu času

```
01 SimpleDateFormat sdf = new SimpleDateFormat("dd-MM-yyyy hh:mm:ss");
02                                                    //formátování času
03 Calendar calendar = Calendar.getInstance();    //kalendář pro práci s časem
04 Random rr = new Random();  //náhodné generování čísel
05
06 for (int i=1; i <= opakuj; i++) {  //cyklus
07    prepinac = i%6; //pro if, při každém průchodu se změní jedna položka data
08
09    if (prepinac == 0) {                    //ukázka, jak se náhodně změní rok
10        calendar.set(Calendar.YEAR, rr.nextInt(56)+2010);
11    //...                                   //pro jistotu omezen (2010 - 2065)
12    }
13    Date time = calendar.getTime(); //převedení kalendáře do potřebného typu
14
15    byte [][] data = reader.dlms.write("0.0.1.0.0.255", time,
16                            DataType.OCTET_STRING, ObjectType.CLOCK, 2);
17        //získání write (SetRequest) zprávy, v tomto případě se nepoužívá
18        //již hotový objekt, ale takto se manuálně vytvoří.
19        //obsahuje LN jméno, hodnotu, typ dat, typ objektu a pozici(atribut)
20
21    reader.readDataBlock(data, reply);      //zaslání DLMS zprávy
22    //...
23 }
```

7.4.5 Opakování AARQ a AARL

Tato metoda je nejjednodušší. Volají se zde již zmíněné metody, které se nachází v každém způsobu generování. Jen zde dochází k jejich neustálému střídání. Prvně je volána metoda `initializeConnection2(...)`, která obsahuje AARQ a ActionRequest zprávy. Následně je zaslána AARL zpráva pomocí `close2()`.

Lze si povšimnout, že obě metody mají v názvu číslo 2. Jsou to jen minimálně upravené jejich originální základy. Především `close2()` zároveň neuzavírá i TCP spojení. Je tak možné udržovat zařízení nedostupné pro ostatní.

7.4.6 AARQ, změna clientID

Tato metoda v cyklu opakovaně navazuje TCP spojení a zasílá AARQ zprávy a následně ukončuje TCP spojení. U posílané AARQ zprávy se s každým průchodem inkrementuje `clientID`. Lze tedy tímto způsobem, při sledování provozu, zjistit jeho správnou hodnotu. V případě špatné hodnoty zařízení neodpovídá vůbec, nebo zašle `rejected-permanent`. V případě správného `clientID` a `serverID` by server měl odpovědět `accepted`. Zbytek autentizace je totiž řešen až v `ActionRequest` zprávách, již popsaných v kapitole 7.3

7.4.7 Špatný ActionRequest

Poslední metoda po úspěšné AARQ zprávě zasílá odpověď na výzvu buď podle specifikace (v případě testovaných zařízeních je také špatná), nebo je vytvořena náhodně. Následně může být spojení ukončeno a postup lze znovu opakovat, nebo zkoušet zasílat zprávy opakovaně bez ukončení spojení. Zařízení však odpoví pouze na první `ActionRequest` zprávu. Na další pokusy již odpovídá zprávou s obsahem `ServiceUnsupported`. Je to z důvodů zamezení možných útoků, kde by se zkoušely různé odpovědi na jednu výzvu. Běžně je výzva dynamická a při každém spojení by se měla měnit. V případě testovaných elektroměrů je však výzva statická.

7.5 Doporučení pro používání generátoru

Vytvořený generátor je dostupný na přiloženém CD jako soubor s příponou .jar nebo jako export projektu z programu Eclipse.

Generátor byl po celou dobu vývoje i jeho testování používán spolu s programem Wireshark, který zasílané zprávy zachytával a bylo tak možné sledovat skutečné dění na síti. Proto je pro jeho používání doporučeno také využívat generátor spolu s programem pro zachytávání zpráv v reálném čase. Samotný generátor totiž ve většině případů vůbec odpovědi nezobrazuje.

V některých případech je dokonce získávání odpovědí úplně ignorováno. Především pro urychlení běhu. Například u generování zpráv typu 2, zamezení spojení bez čekání na odpovědi, může docházet při příliš malém zpoždění k tomu, že se velké množství zpráv nebude ani posílat. Ve výsledku pak nebude odpovídat číslo zaslaných zpráv se skutečným počtem zaslaných zpráv.

8 Výsledky

Během vytváření generátoru byly všechny metody testovány na jediném elektroměru (na adrese `10.10.1.122`). Díky neustálému zasílání testovacích zpráv došlo v elektroměru k určitému druhu výjimečného stavu. Na většinu dotazů na objekty a jejich atributy (větší jak 1) byly přijímány chybové odpovědi. Především se jednalo o `Access Error` několika typů, například:

- „Data Block Unavailable“
- „Device reports a undefined object“
- „Device reports Read-Write denied“
- „Device reports a temporary failure“
- „Device reports a hardware fault“

Dokonce dotazy z koncentrátoru na tyto chybové objekty končily také chybou. Po dalším testování a generování stále dalších zpráv nastala daleko závažnější situace. A to taková, že elektroměr zahazoval veškeré TCP spojení. Nejspíše došlo k uzavření portu 4059. Jediným funkčním spojením byl `ping` a `telnet`.

Na přiloženém CD jsou dostupné záznamy čtení všech objektů pomocí funkce `ReadAll` před poruchou (`ReadAll - před.txt`) a po ní (`ReadAll - po.txt`). Při porovnání těchto dvou souborů lze pozorovat, na kterých objektech docházelo k chybám. Přiložený je i obrázek, na kterém je vidět část porovnání těchto dokumentů (`ReadAll-porovnání.png`).

Po zablokování spojení se při čtení z koncentrátoru objevovala chybová hláška „UnexpectedResponse“. Nachází se také na CD (`UnexpectedResponse, získání času.png`).

Obě chyby je možné vidět na zachyceném spojení přímo na koncentrátoru. Soubory (`Vadné hodnoty.pcap`) a (`Zahazování všech spojení.pcap`).

Na CD jsou k dispozici další soubory obsahující záchyt komunikace při různých metodách. Podle jejich názvu lze poznat, co se na určitém souboru nachází. Ze všech souborů byly odstraněny citlivé informace týkající se přítomných zařízení. Především se jedná o MAC adresy, výzvy a jejich odpovědi a autentizační heslo.

9 Závěr

Bakalářská práce je zpočátku zaměřena na prostudování protokolu DLMS (kapitola 3) a s ním souvisejícího objektového modelu COSEM (kapitola 4), jehož součástí jsou jednotlivé objekty OBIS. Okrajově je zmíněna reálná aplikace protokolu DLMS/COSEM v české distribuční soustavě. Z uvedených zdrojů je patrné, že rozšíření chytrých elektroměrů v České republice je stále na nízké úrovni. Využívají je zejména výrobci a další dodavatelé do sítě. Nejrozšířenější využití je mezi účastníky projektu Smart region Vrchlabí.

Nabyté zkušenosti o protokolu DLMS byly využity k analýze zařízení přítomných v laboratoři a následně k vytvoření aplikace – Zátěžového generátoru DLMS/COSEM zpráv. Během zkoumání těchto zařízení bylo nalezeno hned několik odchylek od samotné specifikace, které jsou zmíněné v kapitole 6.2. Některé odchylky nejsou až tolik závažné např. špatný formát času popsaný v kapitole 6.2.1. Další odchylky jsou spíše bezpečnostní zranitelností. Použití vlastního výpočtu autentizace je určitě plus pro bezpečnost, ale zároveň je i velkým mínusem pro nemožnost spolupráce více zařízení různých výrobců.

Největším bezpečnostním problémem na těchto zařízeních je absence dynamických výzev během autentizace. Tím přímo vybízejí k aplikování útoku typu `reply attack`. Na druhou stranu využití statických výzev umožnilo vytvoření a testování samotného generátoru. Využitá autentizace je typu HLS, ale mohla by používat navíc nějaký doplňkový bezpečnostní algoritmus (kapitola 3.7).

Samotný vytvořený generátor je dostupný na přiloženém CD (včetně zdrojových kódů) a jeho tvorba je popsaná v kapitole 7. Během vývoje bylo dosaženo výjimečného stavu na jednom z elektroměrů (kapitola 8).

Literatura

[1] *DLMS UA* [online]. [cit. 2019-02-22]. Dostupné z: https://www.dlms.com/

[2] *IEC 62056* [online]. IEC Webstore [cit. 2019-02-22]. Dostupné z: https://webstore.iec.ch/searchform&q=IEC%2062056

[3] *ČSN EN 62056* [online]. [cit. 2019-02-22]. Dostupné z: http://www.technicke-normy-csn.cz/technicke-normy/elektrotechnika-35/elektromery-3561

[4] MOLEK, Tomáš. Smart region Vrchlabí. *oEnergetice* [online]. [cit. 2018-12-01]. Dostupné z: http://oenergetice.cz/elektrina/smart-region-vrchlabi-prvni-ceska-chytra-sit/

[5] DLMS – The basis for interoperability. *dlms.com* [online]. Zug, Switzerland: DLMS User Association [cit. 2018-11-10]. Dostupné z: https://www.dlms.com/downloads/161109_dlms_folder_web_singlepage.pdf

[6] SATO, Takuro. *Smart grid standards: specifications, requirements, and technologies* [online]. Singapore, [2015] [cit. 2018-11-24]. ISBN 978-111-8653-692.

[7] Green Book: Architecture and Protocols. *DLMS* [online]. Zug, Switzerland: DLMS User Association, 2017 [cit. 2018-11-13]. Dostupné z: https://www.dlms.com/files/Green-Book-Ed-83-Excerpt.pdf

[8] Blue Book: COSEM Inteface Classes and OBIS Object Identification System. *DLMS* [online]. Zug, Switzerland: DLMS User Association, 2017 [cit. 2018-11-23]. Dostupné z: https://www.dlms.com/files/Blue-Book-Ed-122-Excerpt.pdf

[9] Yellow Book: Conformance Testing Process. *DLMS* [online]. Zug, Switzerland: DLMS User Association, 2017 [cit. 2018-11-23]. Dostupné z: https://www.dlms.com/files/Yellow-Book-Ed-61-Excerpt.pdf

[10] *Flag ID List* [online]. DLMS UA [cit. 2019-02-22]. Dostupné z: https://www.dlms.com/eng/flag-id-list-44143.shtml

[11] JIRKA, Matěj. *Framework DLMS/COSEM pro sběr dat v AMM systémech.* Praha, 2016. Diplomová práce. ČVUT.

[12] LIMPHAPAYOM, Siwarat, Duong Hoang LE a Wanchalerm PORA. A simulation of a communication between a smart meter and a data concentrator unit conformed to DLMS/COSEM upon an HDLC profile. *2014 International Conference on Electronics, Information and Communications (ICEIC)* [online]. IEEE, 2014, 2014, , 1-2 [cit. 2018-11-20]. DOI:

10.1109/ELINFOCOM.2014.6914454. ISBN 978-1-4799-3942-8. Dostupné z: http://ieeexplore.ieee.org/document/6914454/

[13] LURING, Norman, Daniel SZAMEITAT, Stefan HOFFMANN a Gerd BUMILLER. *Analysis of security features in DLMS/COSEM: Vulnerabilities and countermeasures* [online]. IEEE, 2018, 2018, , 1-5 [cit. 2018-11-20]. DOI: 10.1109/ISGT.2018.8403340. ISBN 978-1-5386-2453-1. Dostupné z: https://ieeexplore.ieee.org/document/8403340/

[14] *jDLMS User Guide* [online]. OpenMUC [cit. 2019-02-22]. Dostupné z: https://www.openmuc.org/dlms-cosem/user-guide/

[15] *Wireshark* [online]. Gerald Combs [cit. 2019-02-22]. Dostupné z: https://www.wireshark.org/

[16] *DLMS analysis* [online]. GitHub [cit. 2019-02-22]. Dostupné z: https://github.com/matousp/dlms-analysis

[17] MATOUŠEK, Petr. *Analysis of DLMS Protocol* [online]. FIT-TR-2017-13, Brno, CZ, 2017 [cit. 2018-12-09]. Dostupné z: http://www.fit.vutbr.cz/research/pubs/tr.php?id=11616. Technická zpráva. FIT VUT.

[18] *Gurux DLMS Java* [online]. GitHub [cit. 2019-02-22]. Dostupné z: https://github.com/Gurux/gurux.dlms.java

[19] *RawCap* [online]. Netresec AB [cit. 2018-11-27]. Dostupné z: https://www.netresec.com/?page=rawcap

[20] *Dual licensing of Gurux products* [online]. Gurux [cit. 2019-04-27]. Dostupné z: https://www.gurux.fi/duallicense

[21] *Obecná veřejná licence GNU v.2* [online]. GNU GPL Czech [cit. 2019-05-05]. Dostupné z: http://www.gnugpl.cz/v2/

[22] *Eclipse* [online]. The Eclipse Foundation [cit. 2019-02-22]. Dostupné z: https://www.eclipse.org/downloads/packages/release/2018-12/r

[23] *Java JDK* [online]. Oracle [cit. 2019-02-22]. Dostupné z: https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

[24] *Scene Builder* [online]. Gluon [cit. 2019-02-22]. Dostupné z: https://gluonhq.com/products/scene-builder/

Seznam symbolů, veličin a zkratek

AA	Application Associations
AARE	Application Associations Response
AARQ	Application Associations Request
ACSE	Association Control Service Element
AES	Advanced Encryption Standard
AP	Application Process
APDU	Application Protocol Data Unit
ASE	Application Service Elements
ASO	Application Service Object
COSEM	Companion Specification for Energy Metering
DLMS	Device Language Message Specification
DLMS AL	DLMS Application Layer
DLMS UA	DLMS User Association
GBT	General block transfer
GCM	Galois/Counter Mode
GPRS	General Packet Radio Service
GUI	Graphical User Interface
HDO	Hromadné dálkové ovládání
HLS	High Level Security
ID	Identifier
IP	Internet Protocol
LDN	Logical Device Name
LLS	Low Level Security
LN	Logical Name
LTE	Long-Term Evolution
MAC	Media Access Control
NB-IoT	Narrowband - Internet of Things
OBIS	OBject Identification System
PDU	Protocol Data Unit
SAP	Service Access Point
SN	Short Name
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
UTC	Coordinated Universal Time
VPN	Virtual Private Network

Seznam příloh

A Obsah přiloženého CD

Obsah CD je doplněný o logy a zachycené komunikace. V informačním systému se nachází pouze upravený vytvořený program. Pro veřejnost byl program zbaven citlivých údajů umožňujících připojení k laboratornímu zařízení.

- / ..... kořenový adresář přiloženého CD
  - Generátor ..... Zdrojový kód
    - DLMS_generator.jar ..... Spustitelný generátor
    - generatorDLMS.zip ..... Export projektu z programu Eclipse
    - použitéNástroje.txt
  - Logy a další soubory ..... výpisy různých funkcí, např. ReadAll
    - ReadAll - 3fazovy.txt
    - ReadAll - před.txt
    - ReadAll - po.txt
    - ReadAll-porovnání.png
    - UnexpectedResponse, získání času.png
  - Zachycená komunikace .....Vybraná zachycená komunikace
    - Dlouhý test
      - Záznam z PC.pcap
      - Záznam z SCU.cap
    - Z koncentrátoru ..... Komunikace zachycená na koncentrátoru
      - Vadné hodnoty.pcap
      - Zahazování všech spojení.pcap
      - Aktivní generátor, snaha připojení SCU.pcap
      - Breaker, ActivityCalendar.pcap
      - První záchyt na SCU.pcap
    - Asociace - 0.0.40.0.1.255.pcap
    - Funkce - ReadAll.pcap
    - Pokus - Breaker - SetRequest.pcap
    - První odezva, špatné clientID.pcap
    - Správné ID, špatná výzva.pcap
    - TemporaryFailure.pcap
    - Zamezit typ-2, Žadosti poté odpovědi.pcap
  - Zátěžový_generátor_DLMS-Kohout.pdf ..... Elektronická verze práce